UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis
Mention : Informatique

Présentée et soutenue par

Dorian Mazauric

Optimisation discrète dans les réseaux de télécommunication : reconfiguration du routage, routage efficace en énergie, ordonnancement de liens et placement de données

Thèse dirigée par Jean-Claude Bermond et Philippe Nain

préparée chez Mascotte et Maestro

soutenue le 07 novembre 2011

Rapporteurs :	Augustin Chaintreau	-	Professeur à Columbia University
	Eric Fleury	-	Professeur à l'Ecole Normale Supérieure de Lyon
	Pierre Fraigniaud	-	Directeur de Recherche CNRS au LIAFA
Examinateurs :	David Coudert	-	Chargé de Recherche INRIA
	Alain Jean-Marie	-	Directeur de Recherche INRIA
	Yann Vaxès	-	Professeur à l'Université de la Méditerranée
Directeurs :	Jean-Claude Bermond	-	Directeur de Recherche CNRS
	Philippe Nain	-	Directeur de Recherche INRIA

Table des matières

Chapitre 1

Introduction

Sommaire

Les réseaux de télécommunication ont une place centrale dans les technologies de l'information et de la communication (TIC). Les réseaux optiques sont déployés dans des zones où le trafic est très important. Ils permettent notamment d'interconnecter les différents continents, les pays, les principales villes, les grands centres de données et même d'apporter l'Internet à très haut débit chez les particuliers. Les réseaux sans-fil ont été largement déployés en raison de leurs faibles coûts et de leurs facilités d'installation. Ces derniers sont plus flexibles et permettent un usage supportant la mobilité. De multiples technologies ont été développées pour ce type de réseaux : WiFi, WiMAX, GSM, 3G, faisceaux hertziens, satellites, etc. Depuis une dizaine d'années, les réseaux pair-à-pair ont bouleversé les pratiques d'échange de données entre les utilisateurs. Le problème de la diffusion d'un contenu via un réseau de type pair-à-pair a été largement étudié. Une autre application de ces réseaux de type pair-à-pair est le stockage des données à grande échelle.

Les différents types de réseaux précédemment mentionnés ont tous leurs spécificités, mais de nombreuses problématiques leurs sont communes. En effet, pour tous ces réseaux, il est important d'assurer la meilleure qualité de service possible, de garantir la stabilité du système et de minimiser les ressources nécessaires et donc le coût de fonctionnement.

Pouvoir réagir rapidement et de manière efficace à certaines perturbations dans un réseau est un enjeu majeur car cela assure la stabilité de ce dernier et une bonne qualité de service pour les utilisateurs. Ces perturbations peuvent prendre de multiples aspects : pannes d'un ou de plusieurs équipements du réseau, prise en compte d'un trafic variable. Dans un réseau optique, une panne d'un lien doit être prise en compte par les opérateurs de manière à minimiser les perturbations occasionnées aux utilisateurs du réseau. Les opérateurs doivent également faire face à une possible évolution du trafic. Pour le stockage de données dans les réseaux de type pair-à-pair, il est indispensable de concevoir des systèmes minimisant la probabilité de perdre une donnée en cas d'inactivité définitive d'un ou de plusieurs pairs. Pour ce faire, les mécanismes de placement et de reconstruction doivent être optimisés. L'envoi de données par les utilisateurs en un temps raisonnable est parfois un point critique dans certains types de réseau. En effet, dans les réseaux sans-fil,

certaines transmissions entre différentes paires de nœuds du réseau ne peuvent pas se faire simultanément en raison des interférences. Ainsi, un bon algorithme doit déterminer quels liens du réseau peuvent être activés dans le but d'assurer la stabilité du système et donc une bonne qualité de service pour les utilisateurs notamment en termes de débit et de délai. Cela doit parfois se faire sans connaissance a priori de la dynamique du trafic.

La minimisation de la consommation d'énergie dans les réseaux cœur est un enjeu relativement récent. En effet, le fonctionnement de ce type de réseaux implique une grande dépense d'énergie. Les opérateurs s'y intéressent fortement pour des raisons écologiques mais aussi économiques. Une des questions importantes est d'arriver à minimiser la consommation d'énergie tout en gardant une qualité de service satisfaisante, notamment au niveau de la tolérance aux pannes.

Les moyens à notre disposition pour optimiser ces différents paramètres du réseau sont divers : optimisation du routage, de la reconfiguration du routage, du placement des données, de l'ordonnancement des liens.

Dans un réseau filaire, en cas de panne et/ou de grande dynamique du système, une des réponses naturelles est de modifier le routage. Mais ces modifications peuvent entraîner de grosses perturbations pour les utilisateurs. Le chapitre 2 traite de la manière d'effectuer la reconfiguration du réseau en minimisant les perturbations. Le choix du routage peut également permettre de minimiser les ressources énergétiques utilisées par le réseau. En effet, il a été montré que la consommation du réseau est indépendante de sa charge mais dépend du nombre d'équipements utilisés. Le chapitre 3 concerne la détermination de routages minimisant le nombre d'interfaces utilisées. Dans le chapitre 4, nous nous intéressons aux algorithmes d'ordonnancement des liens (*call scheduling*) dans les réseaux sans-fil prenant en compte les interférences. Enfin dans le chapitre 5, nous considérons le problème de stockage de données dans les réseaux pair-à-pair. Nous étudions l'impact de différentes politiques de placement sur la durée de vie des données, puis déterminons un choix de placement optimal.

Pour résoudre ces problèmes, nous utilisons les outils théoriques des mathématiques discrètes (graphes, configurations, optimisation combinatoire), d'algorithmique (complexité, algorithmique distribuée) et de probabilités.

Dans la suite de l'introduction, nous résumons les principaux résultats que nous avons obtenus.

1.1 Principaux résultats

Reconfiguration du routage dans les réseaux optiques : chapitre 2.

Nous nous intéressons à la reconfiguration du routage des requêtes dans les réseaux optiques à multiplexage en longueurs d'onde. A chaque demande de connexion, il doit être attribué un chemin optique : un chemin dans le réseau et une longueur d'onde. La contrainte de multiplexage fait que deux requêtes passant par une même fibre du réseau, doivent avoir deux longueurs d'onde distinctes. La contrainte de non

conversion traduit le fait qu'une requête doit avoir la même longueur d'onde de bout en bout. Le réseau est modélisé par un multi-digraphe avec les arêtes représentant les différentes fibres.

Lorsque des changements de la topologie physique et/ou de l'ensemble des requêtes de connexion surviennent, il est alors nécessaire de trouver un nouveau routage optique. Par exemple, une opération de maintenance sur un lien du réseau peut prendre quelques dizaines de minutes. Il faut alors trouver un autre routage (n'utilisant plus le lien en maintenance) et rerouter les connexions concernées par ces perturbations. Dans le meilleur des cas, ce reroutage séquentiel des requêtes peut se faire sans aucune interruption des connexions : établir la nouvelle route optique d'une requête et éteindre l'ancienne pour chacune des demandes. Le problème est que, même pour des instances très simples, il est parfois nécessaire d'interrompre une ou plusieurs connexions. Etant donnés un routage courant et un routage final des requêtes, le problème que nous étudions est de trouver un ordre de reroutage des requêtes selon certains critères liés aux interruptions. Nous modélisons chaque instance de ce problème par un digraphe, appelé digraphe de dépendances, et nous établissons l'équivalence entre un reroutage des requêtes et un jeu avec des agents mobiles dans ce digraphe.

Nous étudions deux paramètres importants : le nombre d'interruptions simultanées et le nombre total d'interruptions. Minimiser l'une de ces deux valeurs revient à calculer l'indice de traitement (*process number*) ou l'indice de transmission (cardinalité d'un *minimum feedback vertex set*) du digraphe de dépendances, respectivement.

Nous proposons un algorithme distribué permettant de calculer l'indice de traitement lorsque le digraphe est un arbre orienté symétrique. Pour ce type de digraphes, l'indice de traitement est étroitement lié à l'indice d'échappement sommet (*node search number*), entre autres paramètres. Notre algorithme peut être adapté au calcul de différents paramètres dans les arbres orientés symétriques.

Nous présentons ensuite une étude sur les possibilités de compromis entre le nombre d'interruptions simultanées et le nombre total d'interruptions. Nous introduisons différents paramètres de compromis et montrons que ces derniers sont difficiles à approximer. Entre autres questions, est-il possible de ne pas "trop" augmenter le nombre d'interruptions simultanées (par rapport à la valeur minimum) lorsque le nombre total d'interruptions est minimum ? Nous prouvons que cette dégradation n'est pas bornée pour les différents paramètres. Nous étudions enfin la classe des graphes orientés symétriques, pour laquelle l'augmentation des valeurs de certains paramètres peut être relativement contrôlée, même si certaines questions restent encore ouvertes.

Enfin, nous proposons un modèle permettant de prendre en compte des requêtes prioritaires ne pouvant pas être interrompues. Nous montrons notamment que ce problème se ramène au problème initial avec un digraphe de dépendances transformé. Nous proposons également un modèle avec partage de la longueur d'onde et montrons notamment que déterminer l'existence d'une reconfiguration sans interruption est NP-complet alors que pour le modèle initial, décider cela est très facile. Nous terminons en proposant une troisième extension prenant en compte des

contraintes physiques.

Ce travail a été réalisé avec Sonia Belhareth, Nathann Cohen, David Coudert, Florian Huc, Napoleão Nepomuceno, Nicolas Nisse, Jean-Sébastien Sereni et Issam Tahiri [CMN09c, CMN09b, CHM+09a, CHM11, CM08, CHM08a, CCM+09, CCM+10b, CCM+ar, CCM+10a, CHM+09b, CHM08b].

Routage efficace en énergie : chapitre 3.

L'économie d'énergie dans les réseaux peut être accomplie en utilisant des techniques efficaces de routage ou de conception de réseaux. Nous étudions une architecture simplifiée de réseaux dans laquelle, lorsque deux routeurs sont reliés par un lien, les deux équipements extrémités de ce lien doivent être allumés. Comme il a été montré dans [MSBR09] qu'un équipement a une consommation dépendant plutôt de son activation que de la quantité de trafic, notre objectif pour minimiser la consommation d'énergie est de minimiser le nombre total d'équipements réseaux activés. Autrement dit, ce problème revient à effectuer un routage des demandes en minimisant le nombre d'arêtes du graphe représentant la topologie. Le problème de décider s'il existe un routage valide des demandes est bien connu pour être NP-complet même avec deux demandes. Nous montrons que le problème de calculer ce nombre optimal d'arêtes n'est pas dans APX. Cela reste vrai avec deux demandes ou lorsque les capacités des arêtes sont toutes identiques.

Nous supposons ensuite que les capacités des arêtes sont toutes égales, que les demandes sont de type *all-to-all* ou *one-to-all* et de même volume. Nous étudions un problème orthogonal qui consiste à déterminer le plus petit rapport capacité sur volume des demandes tel qu'il existe un routage valide des demandes utilisant au plus un nombre d'arêtes donné en entrée. Pour des demandes de type *one-to-all*, nous montrons que ce problème est NP-complet. De plus, il n'est pas dans APX si nous ajoutons une contrainte sur le degré des sommets du sous-graphe supportant le routage des demandes.

Pour ces différents problèmes et paramètres, nous donnons des bornes théoriques et des constructions de routages efficaces en énergie pour les graphes généraux, les graphes complets et les grilles. Nous présentons enfin des résultats de simulations pour les grilles et des topologies réelles montrant les économies d'énergie que nous pouvons réaliser ainsi que l'impact de nos solutions sur la longueur des routes et sur la tolérance aux pannes.

Ce travail a été réalisé avec Frédéric Giroire, Joanna Moulierac et Brice Onfroy [GMMO10a, GMMO10b, GMM11].

Ordonnancement des liens dans les réseaux sans-fil : chapitre 4.

Nous considérons dans ce chapitre le problème d'ordonnancement distribué des liens dans les réseaux sans-fil (*call scheduling*). Dans ce type de réseau, certains liens "voisins" ne peuvent pas être activés simultanément, sinon les transmissions interfèrent. Nous considérons des modèles d'interférences binaires, comme ceux utilisés dans [GLS07, BSS09]. Nous supposons que le temps est divisé en étapes et que chacune d'entre elles comporte deux phases distinctes : une phase de contrôle qui détermine quels liens vont être activés et enverront des données durant la seconde

phase. Nous supposons de plus que les arrivées de messages sur chacun des liens du réseau sont aléatoires. Comme les nœuds n'ont pas une connaissance globale du réseau, notre objectif (comme dans [GLS07, BSS09]) est de concevoir pour la phase de contrôle, un algorithme distribué calculant un ensemble de liens n'interférant pas. Pour être efficace, la phase de contrôle doit être aussi courte que possible ; cela est réalisé par des échanges de messages de contrôle durant un nombre constant de *mini-slots* (*overhead* constant).

Nous proposons le premier algorithme entièrement local vérifiant les propriétés suivantes : il est valable quel que soit le modèle d'interférence binaire utilisé ; son nombre de messages de contrôle est constant (indépendant de la taille du réseau et des valeurs des files d'attente associées aux liens du réseau) ; et il ne requiert pas de connaissance particulière de l'état du réseau. En effet, contrairement aux algorithmes existants, nous n'avons pas besoin de connaître les valeurs des files d'attente des liens dans un certain voisinage, une information difficile à obtenir dans un réseau sans-fil avec interférence. Nous prouvons que notre algorithme permet d'obtenir à chaque étape un ensemble maximal de liens actifs (dans chaque zone d'interférence, il y a au moins un lien activé). Nous donnons également des conditions suffisantes de stabilité sous des hypothèses markoviennes. Enfin, les performances (débit, stabilité) de notre algorithme sont étudiées via des simulations, et comparées aux algorithmes existants.

Ce travail a été réalisé avec Jean-Claude Bermond, Vishal Misra et Philippe Nain [BMMN10, BMMN08, BMMN09].

Placement de données dans les réseaux pair-à-pair : chapitre 5.

Nous étudions différents problèmes liés au placement des données dans les réseaux de type pair-à-pair. Les systèmes pair-à-pair à grande échelle représentent un moyen fiable pour stocker des données à faible coût. Les données sont divisées en différents fragments et ces derniers répartis entre les pairs.

Premièrement, nous étudions l'impact des différentes politiques de placement sur la durée de vie des données. Plus particulièrement, nous décrivons des méthodes pour calculer et/ou approximer le temps moyen avant que le système perde une donnée (*Mean Time to Data Loss*). Nous comparons cette métrique pour trois politiques de placement : deux sont locales, distribuant les fragments sur des voisins logiques, et la troisième est globale.

Ensuite, nous considérons un problème légèrement différent (motivé par la réplication des données dans un système de vidéo à la demande [BBJMR09b, JMRBB09]) où chaque donnée est répliquée sur exactement k serveurs et où chaque serveur peut tomber en panne. Nous cherchons à déterminer un placement minimisant la variance du nombre de données indisponibles. Nous ramenons ce problème à un problème d'existence de configurations "équilibrées", problème difficile car il contient le problème de l'existence de systèmes de Steiner.

Ce travail a été réalisé avec Jean-Claude Bermond, Stéphane Caron, Frédéric Giroire, Alain Jean-Marie, Julian Monteiro, Stéphane Pérennes et Joseph Yu [CGM+10b, CGM+10c, CGM+10a, BJMMY11].

Coloration impropre pondérée : annexe A.

Nous étudions un nouveau problème de coloration motivé par un problème pratique d'allocation de fréquences. Dans les réseaux sans-fil, un nœud interfère avec les autres, à un niveau dépendant de nombreux paramètres : la distance entre les nœuds, la topographie physique, les obstacles, etc. Nous modélisons cela par un graphe arête-valué G où le poids d'une arête représente le bruit (ou l'interférence) entre ces deux extrémités. L'interférence totale au niveau d'un nœud est alors la somme de tous les bruits entre ce nœud et les autres nœuds émettant avec la même fréquence. Une k-coloration t-impropre pondérée de G est une k-coloration des nœuds de G (assignation de k fréquences) telle que l'interférence à chaque nœud n'excède pas un certain seuil t. Nous étudions le problème de la coloration impropre pondérée qui consiste à déterminer le nombre chromatique t-impropre pondéré d'un graphe arête-valué G, qui est le plus petit entier k tel que G admette une k-coloration t-impropre pondérée. Nous considérons également le problème *Threshold Improper Colouring* (le problème dual) qui, étant donné un nombre de couleurs (fréquences), consiste à déterminer le plus petit réel t tel que G admette une k-coloration t-impropre pondérée. Nous montrons que tous ces problèmes sont NP-difficiles en présentant tout d'abord des bornes supérieures générales ; en particulier nous prouvons une généralisation du théorème de Lovász pour le problème du nombre chromatique t-impropre pondéré. Nous montrons ensuite comment transformer une instance du problème *Threshold Improper Colouring* en une instance équivalente avec des poids 1 ou M, pour une valeur de M suffisamment grande. Motivé par l'origine du problème, nous étudions un modèle d'interférence particulier pour différentes grilles (carrée, triangulaire, hexagonale) où un nœud produit un bruit d'intensité 1 pour ses voisins et un bruit d'intensité $\frac{1}{2}$ pour les nœuds à distance 2. Le problème consiste alors à déterminer le nombre chromatique t-impropre pondéré lorsque G est le carré d'une grille et que les poids des arêtes valent 1 si les deux nœuds extrémités sont adjacents dans la grille, et $\frac{1}{2}$ sinon. Enfin, nous modélisons le problème par des programmes linéaires en nombres entiers, nous proposons des heuristiques et des algorithmes exacts d'énumération par la technique du *Branch-and-Bound*. Nous les testons pour des graphes aléatoires ressemblant à des réseaux cellulaires, à savoir des tessellations de Poisson-Voronoi.

Ce travail a été réalisé avec Julio Araujo, Jean-Claude Bermond, Frédéric Giroire, Frédéric Havet et Remigiusz Modrzejewski [ABG+11a, ABG+11b].

Distance moyenne dans les réseaux récursifs : annexe B.

Nous étudions deux caractéristiques d'une classe de graphes proposée par Zhang *et al.* dans [ZRG06]. Ces graphes sont construits de manière récursive comme suit. ZRG_0 est le graphe vide et ZRG_1 représente un cycle composé de trois sommets. Pour $t \geq 2$, ZRG_t est construit à partir de ZRG_{t-1} en ajoutant pour chaque arête créée à l'étape t, un nouveau sommet en reliant ce dernier aux deux sommets extrémités de l'arête. Nous prouvons une formule close pour la distance moyenne entre deux paires de sommets de ZRG_t pour tout $t \geq 0$. Nous étudions ensuite cette distance moyenne pour des graphes construits de manière non déterministe.

Ce travail a été réalisé avec Jean-Claude Bermond, Philippe Giabbanelli et Stéphane Pérennes [BGM10, GMP10].

1.2 Publications

Edition d'actes

1. Frédéric Giroire, and Dorian Mazauric, editors. *11es Journées Doctorales en Informatique et Réseaux* (**JDIR 2010**). Sophia Antipolis, France, 24 March - 26 March, 2010

Chapitre

1. Frédéric Giroire, Dorian Mazauric, and Joanna Moulierac. *Sustainable Green Computing : Practices, Methodologies and Technologies. Chapter Energy Efficient Routing by Switching-Off Network Interfaces.* Wen-Chen Hu and Naima Kaabouch, editors. To appear.

Revues internationales

1. David Coudert, Florian Huc, and Dorian Mazauric. *A Distributed Algorithm for Computing the Node Search Number in Trees.* **Algorithmica**. To appear.
2. Nathann Cohen, David Coudert, Dorian Mazauric, Napoleão Nepomuceno, and Nicolas Nisse. *Tradeoffs in process strategy games with application in the WDM reconfiguration problem.* **Theoretical Computer Science**. 412(35) :4675-4687, August 2011.

Conférences internationales

1. Julio Araujo, Jean-Claude Bermond, Frédéric Giroire, Frédéric Havet, Dorian Mazauric, and Remigiusz Modrzejewski. *Weighted Improper Colouring.* In Proceedings of International Workshop on Combinatorial Algorithms (**IWOCA 2011**). University of Victoria, Victoria, British Columbia, Canada, 20 June - 22 June, 2011. Lecture Notes in Computer Science. Springer. To appear.
2. Frédéric Giroire, Dorian Mazauric, Joanna Moulierac, and Brice Onfroy. *Minimizing Routing Energy Consumption : from Theoretical to Practical Results.* In Proceedings of IEEE/ACM International Conference on Green Computing and Communications (**GreenCom 2010**). Hangzhou, China, 18 December - 20 December, 2010.

3. Jean-Claude Bermond, Philippe Giabbanelli, and Dorian Mazauric. *Average path length of deterministic and stochastics recursive networks.* In Proceedings of Second Workshop on Complex Networks (**CompleNet 2010**). Communications in Computer and Information Science (CCIS), Springer-Verlag. Rio de Janeiro, Brazil, 13 October - 15 October, 2010.
4. Stéphane Caron, Frédéric Giroire, Dorian Mazauric, Julian Monteiro, and Stéphane Pérennes. *Data Life Time for Different Placement Policies in P2P Storage Systems.* In Proceedings of Third International Conference on Data Management in Grid and P2P Systems (**GLOBE 2010**). Volume 6265 of Lecture Notes in Computer Science. Pages 75-88. Bilbao, Spain, 1 September - 2 September, 2010.
5. Jean-Claude Bermond, Dorian Mazauric, Vishal Misra, and Philippe Nain. *A Distributed Scheduling Algorithm for Wireless Networks with Constant Overhead and Arbitrary Binary Interference.* In Proceedings of ACM **SIGMETRICS 2010**. Pages 345-346. New York, USA, 14 June - 18 June, 2010.
6. Nathann Cohen, David Coudert, Dorian Mazauric, Napoleão Nepomuceno, and Nicolas Nisse. *Tradeoffs in process strategy games with application in the WDM reconfiguration problem.* In Proceedings of Fifth International Conference on Fun with Algorithms (**FUN 2010**). In P. Boldi and L. Gargano, editors, volume 6099 of Lecture Notes in Computer Science. Springer, pages 121-132. Ischia Island, Italy, 2 June - 4 June, 2010.
7. David Coudert, Dorian Mazauric, and Nicolas Nisse. *On Rerouting Connection Requests in Networks with Shared Bandwidth.* In Proceedings of the DIMAP workshop on Algorithmic Graph Theory (**AGT 2009**). In A. Koster and V. Lozin, editors, volume 32 of Electronic Notes in Discrete Mathematics. Elsevier, pages 109-116. Warwick, United Kingdom, 23 March - 25 March, 2009.
8. David Coudert, Florian Huc, Dorian Mazauric, Nicolas Nisse, and Jean-Sébastien Sereni. *Reconfiguration of the Routing in WDM Networks with Two Classes of Services.* In Proceedings of the 13th IEEE Conference on Optical Network Design and Modeling (**ONDM 2009**). Braunschweig, Germany, 18 February - 20 February, 2009.
9. David Coudert, Florian Huc, and Dorian Mazauric. *Computing and updating the process number in trees.* In Proceedings of the 12th International Conference On Principles Of DIstributed Systems (**OPODIS 2008**). In T. P. Baker, A. Bui and S. Tixeuil, editors, volume 5401 of Lecture Notes in Computer Science. Springer, pages 546-550. Luxor, Egypt, 15 December - 18 December, 2008.
10. David Coudert, Florian Huc, and Dorian Mazauric. *A distributed algorithm for computing and updating the process number of a forest.* In Proceedings of the 22nd International Symposium on Distributed Computing

(**DISC 2008**). In G. Taubenfeld, editor, volume 5218 of Lecture Notes in Computer Science. Springer, pages 500-501. Arcachon, France, 22 September - 24 September, 2008.

Conférences nationales

1. Sonia Belhareth, David Coudert, Dorian Mazauric, Nicolas Nisse, and Issam Tahiri. *Reconfiguration avec contraintes physiques dans les réseaux WDM.* In 13es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2011**). In B. Ducourthial and P. Felber, editors, pages 17-20. Cap Estérel, France, 23 May - 26 May, 2011
2. Frédéric Giroire, Dorian Mazauric, and Joanna Moulierac. *Routage efficace en énergie.* In 13es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2011**). In B. Ducourthial and P. Felber, editors, pages 101-104. Cap Estérel, France, 23 May - 26 May, 2011
3. Stéphane Caron, Frédéric Giroire, Dorian Mazauric, Julian Monteiro, and Stéphane Pérennes. *P2P Storage Systems : Data Life Time for Different Placement Policies.* In 12es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2010**). In M. Gradinariu Potop-Butucaru and H. Rivano, editors, pages 17-20. Belle Dune - Cote d'Opale, France, 31 May - 3 June, 2010.
4. Philippe Giabbanelli, Dorian Mazauric, Stéphane Pérennes. *Computing the average path length and a label-based routing in a small-world graph.* In 12es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2010**). In M. Gradinariu Potop-Butucaru and H. Rivano, editors, pages 47-50. Belle Dune - Cote d'Opale, France, 31 May - 3 June, 2010.
5. Nathann Cohen, David Coudert, Dorian Mazauric, Napoleão Nepomuceno, and Nicolas Nisse. *Tradeoffs in routing reconfiguration problems.* In 12es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2010**). In M. Gradinariu Potop-Butucaru and H. Rivano, editors, pages 119-122. Belle Dune - Cote d'Opale, France, 31 May - 3 June, 2010.
6. David Coudert, Florian Huc, Dorian Mazauric, Nicolas Nisse, and Jean-Sébastien Sereni. *Reconfiguration dans les réseaux optiques.* In 11es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2009**). In A. Chaintreau and C. Magnien, editors, pages 25-28. Carry-Le-Rouet, France, 16 June - 19 June, 2009.
7. Jean-Claude Bermond, Dorian Mazauric, and Philippe Nain. *Algorithmes distribués d'ordonnancement dans les réseaux sans-fil.* In 10es Journées

Doctorales en Informatique et Réseaux (**JDIR 2009**). In Alexandre Caminada, editor, pages 55-60. Belfort, France, 2 February - 4 February, 2009.

8. David Coudert, Florian Huc, and Dorian Mazauric. *Algorithme générique pour les jeux de capture dans les arbres.* In 10es Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (**AlgoTel 2008**). In D. Simplot-Ryl and S. Tixeuil, editors, pages 37-40. Saint-Malo, France, 13 May - 16 May, 2008.

En soumission

1. Julio Araujo, Jean-Claude Bermond, Frédéric Giroire, Frédéric Havet, Dorian Mazauric, and Remigiusz Modrzejewski. *Weighted Improper Colouring.* Research Report RR-7590, INRIA, April 2011. Submitted to Journal of Discrete Algorithms (special issue).
2. Stéphane Caron, Frédéric Giroire, Dorian Mazauric, Julian Monteiro, Stéphane Pérennes. *Surveying Different Placement Policies in P2P Storage Systems.* Research Report RR-7209, INRIA, February 2010. Submitted to Peer-to-Peer Networking and Applications.
3. Frédéric Giroire, Dorian Mazauric, Joanna Moulierac, and Brice Onfroy. *Minimizing Routing Energy Consumption : from Theoretical to Practical Results.* Research Report RR-7234, INRIA, May 2010. To be submitted to a journal.
4. Jean-Claude Bermond, Dorian Mazauric, Vishal Misra, and Philippe Nain. *Distributed Call Scheduling in Wireless Network.* Research Report RR-6763, INRIA, December 2008. To be submitted to a journal.
5. Jean-Claude Bermond, Alain Jean-Marie, Dorian Mazauric, and Joseph Yu. *Well Balanced Designs for Data Placement.* Research Report RR-7725, INRIA, September 2011. To be submitted to a journal.
6. Fedor V. Fomin, Frédéric Giroire, Alain Jean-Marie, Dorian Mazauric, and Nicolas Nisse. *To Satisfy Impatient Web surfers is Hard.* Research Report RR-7740, INRIA, September 2011. Submitted to Latin 2012.

Chapitre 2

Reconfiguration du routage dans les réseaux optiques

Sommaire

2.1 Introduction

Ce travail a été réalisé avec Sonia Belhareth, Nathann Cohen, David Coudert, Florian Huc, Napoleão Nepomuceno, Nicolas Nisse, Jean-Sébastien Sereni et Issam Tahiri [CMN09c, CMN09b, CHM+09a, CHM11, CM08, CHM08a, CCM+09, CCM+10b, CCM+ar, CCM+10a, CHM+09b, CHM08b].

2.1.1 Motivations

Nous considérons des réseaux optiques à multiplexage en longueurs d'onde (*Wavelength Division Multiplexing*, WDM). À chaque demande de connexion, un chemin dans le réseau doit lui être attribué ainsi qu'une longueur d'onde. Le multiplexage dans ces réseaux contraint les requêtes passant par une même fibre du réseau à avoir toutes des longueurs d'onde différentes (contrainte de multiplexage). De plus, nous supposons qu'une connexion doit avoir la même longueur d'onde de bout en bout (contrainte de non conversion). Un routage optique pour un ensemble de connexions consiste alors à déterminer pour chacune des requêtes un chemin dans le réseau et une longueur d'onde respectant les contraintes de multiplexage et de non conversion.

Nous modélisons un réseau WDM par un multi-digraphe $G = (V, E)$ avec V représentant l'ensemble des nœuds du réseau et chaque arête $(u, v) \in E$ représente une fibre d'un lien reliant le nœud u et le nœud v. L'ensemble des W longueurs d'onde du réseau est noté $\Lambda = \{\lambda_1, \ldots, \lambda_W\}$. Nous utiliserons l'ensemble d'entiers (ou de couleurs) $C = \{1, \ldots, W\}$ pour représenter les différentes longueurs d'onde. Soit $\Upsilon = \{d_1, \ldots, d_{|\Upsilon|}\}$ l'ensemble des requêtes de connexion. Une route optique d'une requête $d_i = (u_i, v_i)$ est un *chemin coloré* (P_i, c_i) avec P_i un chemin entre u_i et v_i dans G et $c_i \in C$ une couleur représentant la longueur d'onde utilisée. La contrainte de non conversion exprime le fait que de bout en bout, la connexion d_i utilise la même longueur d'onde $c_i \in C$. Un routage optique de Υ est un ensemble de chemins colorés respectant la contrainte de multiplexage : si deux chemins colorés utilisent le même arc, alors leurs couleurs respectives doivent être différentes.

Problème 1 (Problème de routage et d'affectation de longueur d'onde) *Étant donnés un mutli-digraphe $G = (V, E)$ représentant un réseau optique WDM, un ensemble $C = \{1, \ldots, W\}$ représentant les longueurs d'onde disponibles et un ensemble $\Upsilon = \{d_1, \ldots, d_{|\Upsilon|}\}$ représentant les requêtes de connexion, le problème de routage et d'affectation de longueurs d'onde (*Routing and Wavelength Assignment*, RWA) consiste à déterminer un ensemble de chemins colorés $\{(P_1, c_1), \ldots, (P_{|\Upsilon|}, c_{|\Upsilon|})\}$ tel que $\forall i, j \in \{1, \ldots, \Upsilon\}$, $i \neq j$: si $E(P_i) \cap E(P_j) \neq \emptyset$, alors $c_i \neq c_j$ (contrainte de multiplexage).*

Dans [Cou10], Coudert décrit les différentes méthodes qui semblent être les plus efficaces pour résoudre le Problème 1 : programmation linéaire avec génération de colonnes [JMT07] et méthodes basées sur l'arrondi aléatoire du multiflot fractionnaire [BCLR03, CR02b, CRR03, BCL+03, CR02a].

Lorsque des changements de la topologie physique et/ou de l'ensemble des requêtes surviennent, le problème de la reconfiguration du trafic se pose. En effet, les opérateurs de réseaux optiques doivent parfois planifier des opérations de maintenance sur certains liens du réseau (changement temporaire de la topologie) et/ou faire face à des variations du trafic (changement de l'ensemble des requêtes). Dans de telles situations, il est parfois nécessaire de modifier le routage : les routes optiques d'un certain nombre de requêtes utilisent des liens qui sont/seront indisponibles ou de nouvelles requêtes ne peuvent pas être acceptées sans modifier le routage courant.

Différentes questions se posent alors sur le choix du nouveau routage. Comment calculer le nouveau routage ? Quelle est l'importance à donner au routage courant dans ce calcul ? Comment effectuer la reconfiguration de toutes les connexions du routage courant vers le routage final ?

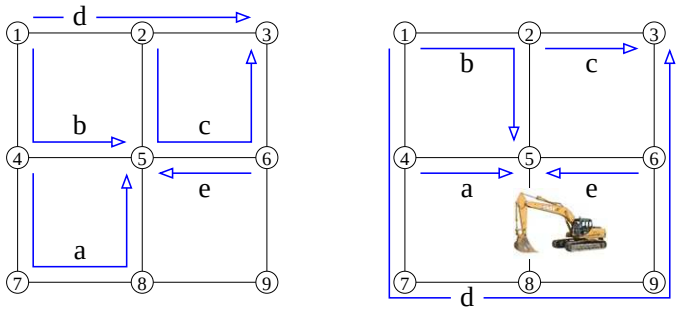
Ces questions liées aux réseaux optiques sont centrales pour tous les réseaux orientés connexions : réseaux téléphoniques dans les années 1970 [Ack79], plus récemment pour les réseaux WDM [LHA94, BEP+96, LL96, MM99, MM99, CBL07, CHM+09a] et MPLS (*Multi-Protocol Label Switching*) [BKP03, JM03, Klo08].

Dans de telles situations, une approche possible est de changer les routes optiques des requêtes séquentiellement jusqu'à atteindre un routage satisfaisant, qui n'utilise pas certaines ressources du réseau. Le changement de route optique d'une requête de connexion peut se faire en utilisant l'opération *Move-To-Vacant* (MTV) [LL96, MM99, CBL07]. Cette opération de migration établit premièrement la nouvelle route optique de la requête concernée tout en préservant la transmission sur l'ancienne durant toute l'opération. Une fois la nouvelle route optique établie, les ressources de l'ancienne deviennent disponibles. Il n'y a donc pas d'*interruption* du trafic lors de cette opération. Notons qu'il est possible d'utiliser l'opération Move-to-Vacant plusieurs fois pour une même requête. Une question importante est alors de savoir si une séquence d'opérations Move-to-Vacant permettant d'atteindre un routage satisfaisant existe toujours ? Quel est le temps de convergence (nombre d'opérations) de cette méthode ? Que faire si une telle séquence n'existe pas ?

Une autre approche investiguée est de calculer un nouveau routage R' en minimisant le nombre de connexions ayant des routes différentes dans le routage courant R et le nouveau R'. Les solutions proposées dans la littérature sont basées sur des programmes linéaires en nombres entiers [BM00, GM03]. En revanche, comme décrit dans [GM03], ces solutions n'apportent pas de réponses pour savoir comment passer de R à R'.

Certains travaux avaient ensuite pour double objectif de calculer le nouveau routage R' et la séquence d'opérations Move-to-Vacant associée. Les solutions utilisent une nouvelle fois des programmes linéaires en nombre entiers (difficiles à résoudre en raison de leurs tailles) et aussi des heuristiques. Voir [BKP03, JM03, BPF04, Klo08] pour les réseaux MPLS et [SL05, ZYWS07] pour les réseaux WDM. Comme précédemment, le problème de l'existence d'une séquence d'opérations Move-to-Vacant pour passer du routage courant R à R' se pose.

Un des problèmes de ces différentes approches est que pour certaines instances, il n'existe pas de séquence d'opérations Move-To-Vacant permettant d'atteindre un routage optique satisfaisant (par exemple un routage optique qui n'utilise pas certaines ressources du réseau qui vont être temporairement indisponibles). Considérons l'instance de la figure 2.1. Chaque lien du réseau représente deux arcs (un dans chaque sens) chacun composé d'une longueur d'onde. Le routage courant R des cinq requêtes a, b, c, d et e est représenté dans la figure 2.1(a). Supposons qu'une opération de maintenance doit être effectuée sur le lien $(5, 8)$ à la suite d'un malencontreux coup de pelleteuse sur ce dernier. Cette opération pouvant prendre quelques dizaines de minutes, il est alors nécessaire de trouver un nouveau routage



(a) Routage R. (b) Routage R'.

FIG. 2.1 – Le réseau est une grille 3×3 avec des liens symétriques chacun composé d'une longueur d'onde. Les figures 2.1(a) et 2.1(b) représentent deux routages, R et R' respectivement, pour l'ensemble de connexions $\{a, b, c, d, e\}$.

optique pour toute cette période. En effet, la route optique de la requête a utilise le lien $(5, 8)$ qui sera indisponible. Il est facile d'observer qu'il n'existe pas de séquence d'opérations Move-To-Vacant permettant d'obtenir un routage optique des cinq requêtes dans le réseau privé du lien $(5, 8)$. En effet, pour chacune des requêtes, il n'existe pas de route optique alternative, en gardant les routes optiques du routage courant R pour toutes les autres requêtes. Cet exemple montre la nécessité de s'autoriser des *interruptions* du trafic lors de la reconfiguration.

Nous nous autorisons ainsi à interrompre la route optique d'une requête avant d'établir sa nouvelle route. Dans ce contexte, il est alors possible de changer la route optique d'une connexion de deux manières (concepts standardisés pour les réseaux MPLS) :

– le *make-before-break* consiste à d'abord établir une nouvelle route optique de la requête et ensuite interrompre la route courante. Cette opération correspond au Move-To-Vacant présenté précédemment et il n'y a donc pas d'interruption de trafic pour cette requête.

– le *break-before-make* consiste à d'abord interrompre la route courante avant d'établir le nouveau chemin optique de la requête. Il y a donc une interruption temporaire de la connexion.

Nous nous focalisons sur la reconfiguration du routage en supposant le routage optique courant et le routage optique final comme étant donnés. Les requêtes sont reroutées une par une et directement sur leurs routes finales respectives.

Problème 2 (Problème de reconfiguration) *Étant donnés une topologie de réseau représenté par un multi-digraphe $G = (V, E)$, un ensemble $C = \{1, \ldots, W\}$ représentant les longueurs d'onde disponibles, un ensemble $\Upsilon = \{d_1, \ldots, d_{|\Upsilon|}\}$ représentant les requêtes de connexion, le routage initial des requêtes R et le routage final des requêtes R', le problème de reconfiguration consiste à déterminer un ordre $\mathcal{O} = (d_1, d_2, \cdots, d_{|\Upsilon|})$ sur les requêtes.*

Pour i allant de 1 à $|\Upsilon|$: si les ressources nécessaires à l'établissement de la nouvelle route optique de la requête d_i sont disponibles, alors un make-before-break est utilisé pour d_i. Sinon sa route courante est interrompue (libérant les ressources correspondantes) et sa nouvelle route sera établie dès que possible (lorsque les ressources nécessaires seront disponibles). Nous établissons les nouvelles routes optiques des requêtes précédemment interrompues lorsque cela est possible.

Pour résumer, un ordre sur les requêtes induit une séquence d'opérations élémentaires. Une opération élémentaire est soit un make (établir une nouvelle route) soit un break (interrompre une route courante). Un make-before-break est utilisé pour une requête d_i dans cette séquence, si le make correspondant à d_i se situe avant le break correspondant à d_i. Sinon un break-before-make est utilisé pour d_i.

Des questions importantes se posent alors. Quel ordre choisir ? Quels sont les paramètres importants liés aux interruptions ?

Considérons l'instance de la figure 2.1. Chaque lien du réseau représente deux arcs (un dans chaque sens) chacun composé d'une longueur d'onde. Le routage initial R des cinq requêtes a, b, c, d et e est représenté dans la figure 2.1(a). Supposons qu'une opération de maintenance doit être effectuée sur le lien $(5, 8)$. Le routage R' décrit dans la figure 2.1(b) est le routage final. Le problème de reconfiguration consiste alors à rerouter les requêtes a, b, c et d (la requête e ne change pas de chemin optique) une par une et directement sur leurs nouvelles routes respectives. Ètant donné que la capacité est unitaire pour chacun des arcs (unique longueur d'onde), il est facile d'observer qu'il est impossible de rerouter les quatre requêtes sans aucune interruption. En effet, aucune requête ne peut être reroutée sans en interrompre une autre préalablement.

Plus précisément, il est impossible d'établir la nouvelle route de la requête
- a car la requête b utilise l'arc $(4, 5)$;
- b car la requête c utilise l'arc $(2, 5)$ et la requête d utilise l'arc $(1, 2)$;
- c car la requête d utilise l'arc $(2, 3)$;
- d car la requête a utilise les arcs $(4, 7)$ et $(7, 8)$, la requête b utilise l'arc $(1, 2)$ et la requête c utilise l'arc $(6, 3)$.

Dans notre exemple, il est donc nécessaire de débuter la reconfiguration par l'interruption d'une requête afin de libérer des ressources (c'est-à-dire d'utiliser un break-before-make pour cette requête).

Soit l'ordre $\mathcal{O} = (d, c, b, a)$. La séquence d'opérations élémentaires est :
- interrompre la requête d (figure 2.4(b)) ;
- établir la nouvelle route de la requête c (figure 2.4(c)) ;
- interrompre l'ancienne route de la requête c (figure 2.4(c)) ;
- établir la nouvelle route de la requête b (figure 2.4(d)) ;
- interrompre l'ancienne route de la requête b (figure 2.4(d)) ;
- établir la nouvelle route de la requête a (figure 2.4(e)) ;
- interrompre l'ancienne route de la requête a (figure 2.4(e)) ;
- établir la nouvelle route de la requête d (figure 2.4(f)).

En résumé, un break-before-make est utilisé pour la requête d et un make-before-break est utilisé pour les requêtes a, b et c.

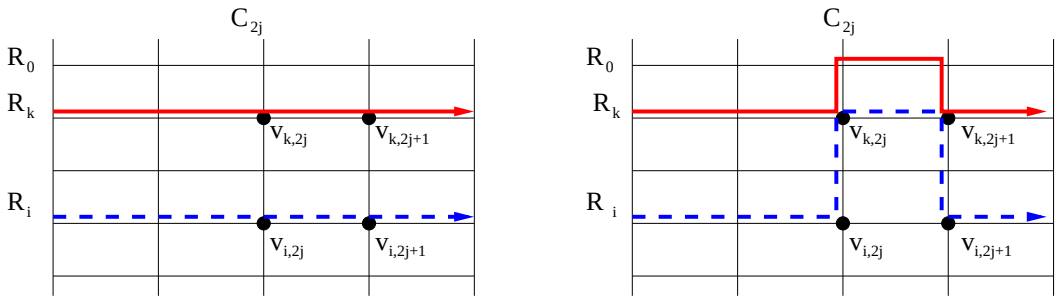


FIG. 2.2 – Schéma de la transformation décrite dans la preuve du Théorème 1.

2.1.2 Digraphes de dépendances

Nous modélisons une instance du problème de reconfiguration en utilisant la notion de digraphe de dépendances introduite initialement dans [JS03]. Soit I une instance du problème. Le digraphe de dépendances $D = (V, A)$ associé à I est tel que nous associons à chacune des requêtes devant changer de route, un sommet $u \in V$. Il y a un arc $(u, v) \in A$ si et seulement si la nouvelle route de la requête associée au sommet u utilise des ressources utilisées par la requête associé à v dans sa route initiale. Dans ce cas, pour rerouter la requête u vers sa nouvelle route, il est nécessaire de rerouter ou d'interrompre v préalablement.

Remarquons qu'à une instance donnée du problème de reconfiguration correspond un unique digraphe de dépendances. En revanche, un digraphe peut représenter différentes instances du problème. Nous prouvons le Théorème 1, motivant l'étude du problème de reconfiguration sans restriction préalable parmi tous les digraphes possibles.

Théorème 1 *Tout digraphe D est le digraphe de dépendances d'une instance du problème de reconfiguration dont le réseau est une grille.*

Preuve : Considérons une topologie de grille et un ensemble de requêtes ayant pour route optique une ligne de la grille dans le routage courant. Si deux connexions i et k sont liées par un arc (i, k) dans le digraphe de dépendances, alors nous construisons les routes optiques des deux connexions comme représentées dans la figure 2.2 afin de créer la dépendance désirée. Remarquons que la route optique de la connexion k est déportée sur une ligne additionnelle, c'est-à-dire une ligne de la grille qui ne correspond à aucune connexion. Pour chaque arc du digraphe de dépendances, nous utilisons une colonne différente de la grille. Les transformations peuvent donc se faire de manière indépendante.

Formellement, soit le digraphe $D = (V, A)$ avec $V = \{c_1, \cdots, c_n\}$ et $A = \{a_1, \cdots, a_m\}$. Nous définissons le réseau G comme une grille $(n + 2) \times (2m)$ telle que chaque arête a capacité 1. Soit R_i la i^e ligne de G $(0 \leq i \leq n + 1)$ et C_i sa i^e colonne $(1 \leq j \leq 2m)$ et soit $v_{i,j} \in V(G)$ le sommet dans $R_i \cap C_j$. Pour tout i,

$1 \leq i \leq n$, la connexion i, correspondant à c_i dans D, est définie entre $v_{i,1} \in V(G)$, le sommet le plus à gauche de R_i et $v_{i,2m} \in V(G)$, le sommet le plus à droite de R_i. La route optique initiale de la requête i suit R_i. Nous présentons maintenant une méthode itérative pour construire la route optique finale de chaque requête. Initialement, pour tout i, $1 \leq i \leq n$, la nouvelle route optique P_i^0 de la requête i est sa route initiale R_i. Après la $(j-1)^e$ étape $(0 < j \leq m)$ de la méthode, soit P_i^{j-1} la valeur courante de la nouvelle route optique de la requête i et supposons que dans le sous-graphe de G induit par les colonnes $(C_{2j-1}, \cdots, C_{2m})$, P_i^{j-1} est égal à R_i. Considérons $a_j = (c_i, c_k) \in A$ et effectuons la transformation suivante décrite dans la figure 2.2. Pour tout $\ell \notin \{i, k\}$, $P_\ell^j = P_\ell^{j-1}$. Ainsi, P_i^j est défini en remplaçant l'arête $(v_{i,2j-1}, v_{i,2j})$ dans P_i^{j-1} par le plus court chemin de $v_{i,2j-1}$ à $v_{k,2j-1}$ (suivant C_{2j-1}), l'arête $(v_{k,2j-1}, v_{k,2j})$ et le plus court chemin de $v_{k,2j}$ à $v_{i,2j}$ (suivant C_{2j}). De manière analogue, P_k^j est défini en remplaçant l'arête $(v_{k,2j-1}, v_{k,2j})$ dans P_k^{j-1} par le plus court chemin de $v_{k,2j-1}$ à $v_{n+1,2j-1}$ si $i < k$ (à $v_{0,2j-1}$ si $i > k$, respectivement), l'arête $(v_{n+1,2j-1}, v_{n+1,2j})$ ($(v_{0,2j-1}, v_{0,2j})$, respectivement) et le plus court chemin de $v_{n+1,2j}$ à $v_{k,2j}$ (de $v_{0,2j}$ à $v_{k,2j}$, respectivement). Il est facile de vérifier que la grille G, l'ensemble initial de routes optiques $\{R_1, \cdots, R_n\}$ et l'ensemble final de routes optiques $\{P_1^m, \cdots, P_n^m\}$ admet D comme digraphe de dépendances. □

Nous montrons dans la Section 2.1.3 que le problème de déterminer une reconfiguration (un ordre sur les requêtes de connexion) pour une instance I est équivalent au problème de déterminer une stratégie de traitement pour le digraphe de dépendances D associé à I.

2.1.3 Stratégies de traitement

Le Théorème 1 montre la pertinence d'étudier le problème de reconfiguration via la notion de digraphe de dépendances. Nous introduisons un *jeu avec des agents mobiles* sur le digraphe de dépendances qui consiste à placer et retirer ces agents selon certaines règles. En effet, le jeu avec des agents consiste à *traiter* tous les sommets du digraphe de dépendances $D = (V, A)$ en utilisant les trois règles suivantes.

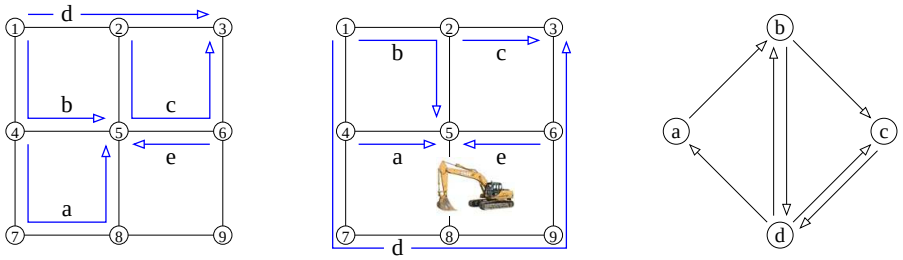
R_1 Placer un agent sur un sommet $v \in V$.

R_2 Supprimer un agent d'un sommet $v \in V$ si chacun de ses voisins sortants est soit déjà traité soit occupé par un agent. Traiter v.

R_3 Traiter un sommet inocuppé $v \in V$ si chacun de ses voisins sortants est soit déjà traité soit occupé par un agent.

Le digraphe D est *traité* si tous ses sommets sont traités. Une séquence d'opérations permettant de traiter tous les sommets de D, et donc de traiter D, est une *stratégie de traitement*. Remarquons que lors d'une stratégie de traitement, un agent retiré d'un sommet traité peut être réutilisé. Un sommet est *couvert* par une stratégie de traitement si un agent a été plaçé sur ce dernier au cours de la stratégie.

Nous montrons l'équivalence entre une reconfiguration des requêtes pour une instance I et une stratégie de traitement pour le digraphe de dépendances D associé à I. En effet, les trois règles précédentes pour le jeu avec des agents correspondent



(a) Routage R. (b) Routage R'. (c) Digraphe de dépendances D.

FIG. 2.3 – Le réseau est une grille 3×3 avec des liens symétriques chacun composé d'une longueur d'onde. Les figures 2.1(a) et 2.1(b) représentent le routage initial R et le routage final R' de l'ensemble de connexions $\{a, b, c, d, e\}$, respectivement. La figure 2.3(c) représente le digraphe de dépendances de cette instance.

d'un point de vue de la reconfiguration des requêtes de connexion à, respectivement :

R_1 Interrompre la connexion de la requête associée à v.
(break d'un break-before-make).

R_2 Établir la connexion (sur la nouvelle route) de la requête associée à v précédemment interrompue car les ressources sont disponibles.
(make d'un break-before-make).

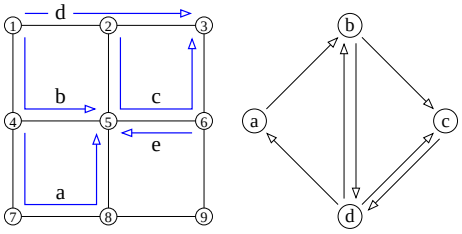
R_3 Établir la connexion (sur la nouvelle route) de la requête associée à v car les ressources sont disponibles (et interrompre l'ancienne route).
(make-before-break).

Ainsi, à un ordre sur les requêtes de connexion d'une instance I correspond une stratégie de traitement pour D, et réciproquement.

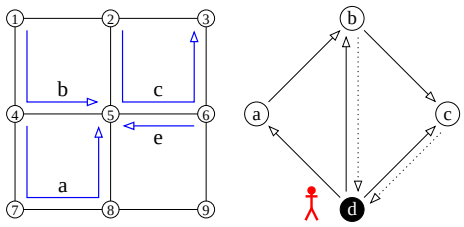
Décrivons le digraphe de dépendances $D = (V, A)$ associé à l'instance de la figure 2.3. Nous avons $V = \{a, b, c, d\}$ car la requête e ne change pas de route. Il y a un arc $(b, c) \in A$ car la requête b utilise, dans sa nouvelle route, des ressources utilisées par la route initiale de la requête c (arc $(2, 5)$ du réseau). Il est donc impératif de rerouter ou d'interrompre la requête c avant d'établir la nouvelle route de b. Nous construisons les autres arcs de D de manière analogue. D est représenté dans la figure 2.3(c).

Décrivons à présent la stratégie de traitement pour D correspondant à la reconfiguration présentée précédemment pour cette instance.

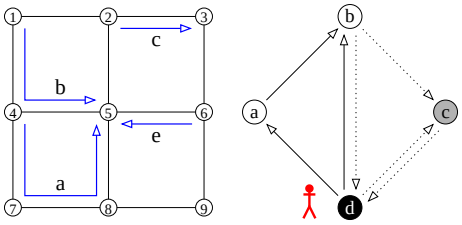
– règle R_1 : interrompre la requête d $\Leftrightarrow$ placer un agent sur d (figure 2.4(b)) ;
– règle R_3 : établir la nouvelle route de la requête c puis interrompre l'ancienne $\Leftrightarrow$ traiter le sommet inocuppé c car son unique voisin sortant d est occupé par un agent (figure 2.4(c)) ;
– règle R_3 : établir la nouvelle route de la requête b puis interrompre l'ancienne $\Leftrightarrow$ traiter le sommet inocuppé b car son voisin sortant c est traité et son voisin



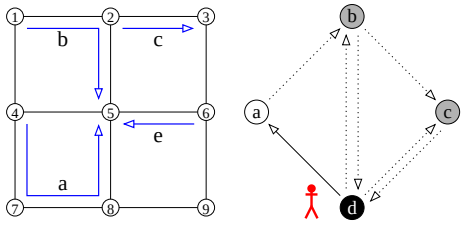
(a) Routage initial R et digraphe de dépendances D de l'instance de la figure 2.3.



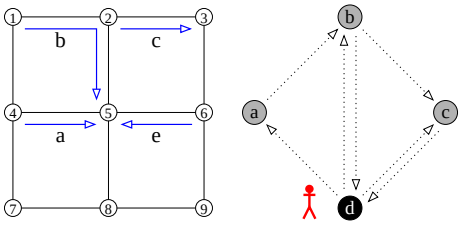
(b) Interrompre la requête d ⇔ Placer un agent sur le sommet d.



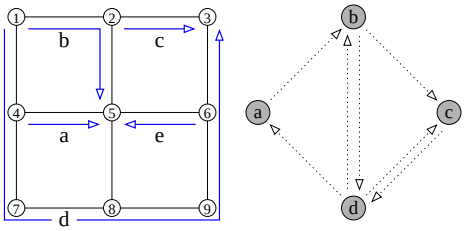
(c) Rerouter la requête c sans interruption ⇔ Traiter le sommet c.



(d) Rerouter la requête b sans interruption ⇔ Traiter le sommet b.



(e) Rerouter la requête a sans interruption ⇔ Traiter le sommet a.



(f) Router la requête d prélablement interrompue ⇔ Retirer l'agent du sommet d et traiter d.

FIG. 2.4 – Soit D le digraphe de dépendances de l'instance I de la figure 2.3. Cette figure représente la reconfiguration de I et la stratégie de traitement de D correspondante. Les sommets traités sont en gris et ceux couverts par un agent en noir. Tous les sommets de la figure 2.4(b) ont au moins un voisin sortant. Il faut donc placer un agent sur un sommet. Nous choisissons de placer un agent sur le sommet d (figure 2.4(b)). Cela correspond à interrompre la requête d (break d'un break-before-make). L'unique voisin sortant du sommet c est alors traité, nous pouvons donc traiter c (figure 2.4(c)). Cela correspond à utiliser un make-before-break pour la requête c. Ensuite, nous pouvons traiter le sommet b (figure 2.4(d)), puis le sommet a (figure 2.4(e)). Nous utilisons un make-before-break pour les deux requêtes correspondantes. Finalement, nous pouvons traiter le sommet d (figure 2.4(f)) et retirer l'agent. Autrement dit, nous établissons la nouvelle route de la requête d préalablement interrompue (make d'un break-before-make).

sortant d est occupé par un agent (figure 2.4(d)) ;

– règle R_3 : établir la nouvelle route de la requête a puis interrompre l'ancienne $\Leftrightarrow$ traiter le sommet inocuppé a car son unique voisin sortant b est traité (figure 2.4(e)) ;

– règle R_2 : établir la nouvelle route de la requête d $\Leftrightarrow$ supprimer l'agent du sommet d car ses trois voisins sortants a, b et c sont traités. Traiter d (figure 2.4(f)).

Dans la Section 2.1.4, nous présentons des résultats existants concernant des paramètres liés au problème de reconfiguration que nous étudierons au cours de ce chapitre. Plus précisément, nous décrivons quelques travaux reliés à la minimisation du nombre total d'interruptions et à la minimisation du nombre maximum d'interruptions simultanées.

2.1.4 État de l'art

Nous présentons dans cette section, une partie des travaux existants concernant des paramètres importants pour le problème de reconfiguration que nous étudions et les stratégies de traitement des digraphes correspondantes.

2.1.4.1 Reconfiguration des requêtes sans aucune interruption

Il existe une reconfiguration sans interruption pour une instance I si et seulement si le digraphe de dépendances D associé à I est sans circuit (*directed acyclic graph*, DAG). En d'autres termes, Il existe une stratégie de traitement pour D sans utiliser d'agent si et seulement si D est sans circuit.

En effet, il est possible de traiter tous les sommets de D en utilisant uniquement la règle R_3 et en suivant un ordre d'effeuillage des sommets de D. Un ordre d'effeuillage consiste à d'abord considérer les feuilles (sommets sans voisin sortant) et se poursuit ainsi récursivement.

De plus si D contient un circuit $(u_0, u_1, \ldots, u_{k-1}, u_k, u_0)$, alors il est facile d'observer qu'un agent est nécessaire. Supposons qu'il existe une stratégie de traitement sans agent pour D, c'est-à-dire une stratégie de traitement qui utilise uniquement la règle R_3. Pour traiter u_0, nous devons préalablement traiter u_1 car $(u_0, u_1) \in A$. Pour traiter u_1, nous devons traiter u_2 car $(u_1, u_2) \in A$. Et ainsi de suite jusqu'à observer que pour traiter u_k, nous devons traiter u_0 car $(u_k, u_0) \in A$. Pour résumer, pour traiter u_0, il est nécessaire de traiter u_0 préalablement. Une contradiction.

Si une instance du problème de reconfiguration admet un digraphe de dépendances D sans circuit, alors l'ordre sur les requêtes induit par tout ordre d'effeuillage des sommets de D, donne une reconfiguration des requêtes sans aucune interruption.

2.1.4.2 Minimisation du nombre total d'interruptions

Minimiser le nombre total de requêtes interrompues au cours de la reconfiguration revient à minimiser le nombre total de sommets couverts par un agent au cours d'une stratégie de traitement du digraphe de dépendances D associé.

De la caractérisation des digraphes pour lesquels il existe une stratégie de traitement sans agent, nous déduisons que pour toute stratégie de traitement pour $D = (V, A)$, l'ensemble $F \subseteq V$ des sommets couverts par un agent forme un *feedback vertex set* de D. Un ensemble F de sommets de D est un feedback vertex set (FVS) si le digraphe induit par les sommets $V \setminus F$ est sans circuit. Un *minimum feedback vertex set* (MFVS) est un FVS de cardinalité minimum. Ainsi, minimiser le nombre de sommets couverts revient à déterminer un MFVS pour D. Cette valeur est l'indice de transmission de D.

Définition 1 (indice de transmission, mfvs) *L'indice de transmission d'un digraphe D, noté $mfvs(D)$, est le plus petit nombre de sommets couverts par une stratégie de traitement pour D.*

Théorème 2 ([Kan92]) *Le problème de calculer l'indice de transmission d'un digraphe est NP-complet et APX-difficile.*

En revanche, déterminer si ce problème est dans APX ou non, est ouvert. Dans [LJ99], les auteurs proposent des opérations de réduction en temps polynomial pour réduire la taille du graphe, avant de proposer un algorithme exponentiel exact.

Pour terminer cette section, il est facile d'observer que le problème de déterminer l'indice de transmission d'un digraphe D revient à déterminer l'indice de transmission de chacune de ses composantes fortement connexes.

Théorème 3 *Étant donné un digraphe D, nous notons $\{SCC_i\}$ l'ensemble des composantes fortement connexes de D. Alors $mfvs(D) = \sum_i mfvs(SCC_i)$.*

Preuve : Tous les sommets d'un circuit de D sont, par définition de l'ensemble $\{SCC_i\}$, dans une seule composante fortement connexe. Ainsi, trouver le nombre minimum de sommets à supprimer pour rendre D sans circuit, revient à trouver le nombre minimum de sommets à supprimer pour rendre chacune des composantes SCC_i sans circuit. □

2.1.4.3 Minimisation du nombre maximum d'interruptions simultanées

Étant donnée une stratégie de traitement pour un digraphe de dépendances D, le nombre maximum de requêtes simultanément interrompues au cours de la reconfiguration correspondante, est le nombre maximum d'agents positionnés sur des sommets de D au cours de la stratégie, et donc le nombre d'agents utilisés. Le problème de minimiser le nombre maximum d'interruptions simultanées revient à calculer l'indice de traitement du digraphe de dépendances [CPPS05].

Définition 2 (p-stratégie de traitement, p-process strategy) *Une p-stratégie de traitement pour un digraphe D est une stratégie de traitement pour D qui utilise au plus p agents.*

Définition 3 (indice de traitement, process number, pn) *L'indice de traitement d'un digraphe D, noté $pn(D)$, est le plus petit p tel qu'il existe une p-stratégie de traitement pour D.*

Dans [CPPS05], les auteurs ont relié l'indice de traitement à la notion de *sommet séparation* [DPS02].

Définition 4 (sommet séparation, *vertex separation*, vs) *Un arrangement linéaire d'un graphe orienté $D = (V, A)$ est une bijection $L : V \to \{1, \ldots, |V|\}$. La coupe maximum induite par les arcs retour de l'arrangement L est mesurée par $\max_{1 \leq i \leq |V|} |M(i)|$ avec $M(i) := \{v \in V : L(v) > i \text{ et } \exists u \in N(v) : L(u) \leq i\}$.*
La sommet séparation de D, notée $\texttt{vs}(D)$, est le minimum pris sur tous les arrangements : $\texttt{vs}(D) = \min_L \max_{1 \leq i \leq |V|} |M(i)|$.

Proposition 1 ([CPPS05]) *Pour tout digraphe D, $\texttt{vs}(D) \leq \texttt{pn}(D) \leq \texttt{vs}(D) + 1$.*

Comme le problème de déterminer la sommet séparation d'un digraphe ne peut pas être approché à une constante additive près [DKL87], à moins que P=NP, alors

Corollaire 1 *Le problème de déterminer l'indice de traitement d'un digraphe ne peut pas être approché à une constante additive près en temps polynomial, à moins que P=NP.*

Dans [Sol09], Solano propose de diviser le problème de calculer l'indice de traitement d'un digraphe D en deux sous-problèmes :

– (P_1) déterminer le sous-ensemble de sommets de D pour lesquels un agent sera placé ;

– (P_2) calculer l'ordre optimal sur ces sommets.

Dans [Sol09], Solano conjecturait que la complexité du problème résidait dans la première phase et que la seconde pouvait être résolue ou approchée par un algorithme en temps polynomial. Nous prouvons dans le Théorème 4 que cela est faux.

Théorème 4 *Le problème de calculer l'indice de traitement d'un digraphe D est NP-complet et n'est pas approximable à une constante additive près en temps polynomial, à moins que P=NP, même si le sous-ensemble de sommets qui peuvent être couverts est donné.*

Preuve : Soit $D = (V, A)$ un digraphe orienté symétrique avec $V = \{u_1, \ldots, u_n\}$. Soit $\hat{D} = (V', A')$ un digraphe orienté symétrique avec $V' = V \cup \{v_1, \ldots, v_n\}$ et $\hat{D}$ est obtenu à partir de D en ajoutant deux arcs symétriques entre u_i et v_i pour $i = 1, \ldots, n$. Il est facile d'observer qu'il existe une stratégie de traitement optimale pour $\hat{D}$ telle que l'ensemble des sommets occupés est V. En effet, notons que, pour tout i, au moins un des deux sommets u_i ou v_i doit être couvert par un agent (tout FVS de D contient au moins un des deux sommets v_i ou u_i). De plus, si une étape de la stratégie de traitement pour $\hat{D}$ consiste à placer un agent sur le sommet v_i,

alors la stratégie de traitement peut être facilement transformée en posant un agent sur u_i à la place.

Considérons le problème du calcul d'une stratégie de traitement optimale pour $\hat{D}$ si l'ensemble des sommets couverts par les agents est contraint d'être V. Par la remarque précédente, une telle stratégie optimale existe toujours. Il est alors facile de vérifier que ce problème est équivalent à celui de calculer l'indice d'échappement sommet (voir la section 2.1.4.4), et donc la pathwidth, du graphe non dirigé sous-jacent de D qui est NP-complet [MHG+88] et qui n'est pas approximable à une constante additive près en temps polynomial [DKL87], à moins que P=NP. □

Il est possible de remarquer que l'indice de traitement d'un digraphe D est l'indice de traitement maximum parmi ses composantes fortement connexes.

Théorème 5 ([CS11]) *Étant donné un digraphe D, nous notons $\{SCC_i\}$ l'ensemble des composantes fortement connexes de D. Alors $pn(D) = \max_i\{pn(SCC_i)\}$.*

Dans [CS11], Coudert et Sereni ont caractérisé les digraphes d'indice de traitement au plus 2. Soit $D = (V, A)$ un digraphe avec $n = |V|$ et $m = |A|$. Rappelons qu'il existe une 0-stratégie de traitement si et seulement si D est sans circuit. Rappelons que l'indice de traitement de D est l'indice de traitement maximum parmi toutes les composantes fortement connexes de D (Théorème 5). Supposons, sans perte de généralité, que D est fortement connexe et qu'il n'existe pas de 0-stratégie de traitement pour D. Il existe une 1-stratégie de traitement pour D si, et seulement si, il existe un sommet $u \in V$ tel que $D - \{u\}$ est sans circuit (DAG). Les auteurs mettent en exergue un algorithme avec une complexité en temps et en espace de $O(n + m)$. Enfin, ils caractérisent les digraphes d'indice de traitement 2 et proposent un algorithme en $O(n(n + m))$ pour la reconnaissance de tels digraphes.

2.1.4.4 Digraphes orientés symétriques

Avant de présenter différents paramètres liés à l'indice de traitement dans les graphes orientés symétriques et leurs relations mutuelles, nous exposons quelques remarques concernant le calcul de l'indice de transmission dans de tels digraphes.

Soit D un digraphe orienté symétrique. Le problème de calculer $mfvs(D)$ est équivalent au problème de couverture minimum de sommets (*minimum vertex cover*) dans le graphe non-orienté sous-jacent. En effet, chaque paire d'arcs entre deux sommets $u \in V(D)$ et $v \in V(D)$ forme un circuit. Ainsi, dans le graphe non-orienté sous-jacent, le problème revient à déterminer le plus petit ensemble de sommets couvrant chacune des arêtes. Le problème de couverture minimum de sommets est NP-complet même pour la classe des graphes cubiques [GJ79] et pour la classe des graphes planaires de degré au plus 3 [GJ77]. De plus, le problème de calculer une couverture minimum de sommets est APX-complet et il a été prouvé dans [DS04], qu'il n'existait pas d'algorithme d'approximation en temps polynomial à un facteur au plus 1.3606, à moins que P=NP.

Dans la suite de la section, nous présentons tout d'abord la notion d'indice d'échappement sommet, l'homologue algorithmique de la notion de pathwidth introduite par Robertson et Seymour [RS83]. Cet indice est défini en termes de jeu des gendarmes et du voleur (*graph searching problem*) [Bre67, DPS02, FT08, KP86, Par78]. Nous présentons ensuite l'indice d'échappement arête avant de (re)définir l'indice de traitement en termes de jeu de gendarmes et de voleur.

Indice d'échappement sommet. L'indice d'échappement sommet est le nombre minimum d'agents nécessaires à la capture d'un fugitif invisible et arbitrairement rapide caché dans un graphe. Les règles de ce jeu à deux joueurs sont les suivantes : à chaque étape, le premier joueur peut déplacer le fugitif d'un sommet à un autre via un chemin dans lequel aucun sommet n'est couvert par un agent. Le second joueur peut alors effectuer une des deux actions suivantes :

(1) placer un agent sur un sommet ;

(2) enlever un agent d'un sommet.

Le fugitif est capturé lorsqu'un agent est situé sur le même sommet. Rappelons que le second joueur ne connaît pas la position du fugitif (invisible).

Définition 5 (p-stratégie de capture, p-search strategy) *Une p-stratégie de capture pour un graphe G est une stratégie de capture pour G qui utilise au plus p agents.*

Définition 6 (indice d'échappement sommet, node search number, ns) *L'indice d'échappement sommet d'un graphe G, noté $\mathtt{sn}(G)$, est le plus petit p tel qu'il existe une p-stratégie de capture pour G.*

Par exemple, une étoile a un indice d'échappement sommet de 2, un chemin a un indice d'échappement sommet de 2, un cycle a un indice d'échappement sommet de 3 et une grille $n \times n$, avec $n \geq 2$, a un indice d'échappement sommet de $n + 1$.

Théorème 6 ([KP86]) *Le problème de calculer l'indice d'échappement sommet d'un graphe est NP-difficile.*

Tout au long d'une p-stratégie de capture, les sommets peuvent être divisés en trois catégories : les sommets *gardés* par des agents, les sommets *non sécurisés* sur un desquels peut se trouver le fugitif et les sommets *sécurisés* où le fugitif ne peut pas être.

Définition 7 (p-stratégie de capture monotone) *Une p-stratégie de capture pour un graphe G est* monotone *si la partie non sécurisée de G n'augmente jamais au cours de cette stratégie. En d'autres termes, un sommet qui a été gardé par un agent, ne peut jamais abriter le fugitif après le retrait de l'agent de ce sommet.*

LaPaugh a prouvé dans [LaP93] qu'il est possible de considérer uniquement des stratégies monotones.

Théorème 7 ([LaP93]) *Si une p-stratégie de capture pour un graphe G existe, alors il existe une p-stratégie de capture monotone pour G.*

Une p-stratégie de capture monotone pour $G = (V, E)$ est une séquence de $2n = 2|V|$ actions des p agents : $m_1, \ldots, m_{2n}$, avec m_i, $1 \leq i \leq 2n$, une des deux actions suivantes *placer un agent sur un sommet* $u \in V$ et *enlever un agent d'un sommet* $u \in V$. Remarquons qu'un agent peut être enlevé d'un sommet si, et seulement si, il a été placé préalablement au cours de la stratégie. De plus, nous considérons uniquement des stratégies qui ne placent pas plus d'un agent par sommet. Ainsi, nous supposons qu'un agent n'est jamais placé sur un sommet gardé ou sécurisé. En résumé, nous considérons uniquement des stratégies avec un nombre n d'actions et où tous les agents sont enlevés du graphe à la fin de la stratégie.

Avant de présenter les liens entre l'indice d'échappement sommet, la pathwidth et la sommet séparation, il est intéressant de citer la notion d'*indice d'échappement connexe* (*connected search number*), qui est similaire à celle de l'indice d'échappement sommet. En effet, la seule différence est que la partie traitée du graphe, dans laquelle le fugitif ne peut pas être, doit être un sous-graphe connexe. Pour les arbres, il a été montré que ce paramètre est au plus deux fois la valeur de l'indice d'échappement sommet [BFST03]. Un algorithme distribué linéaire est proposé dans [BFFS02].

L'indice d'échappement sommet est l'homologue algorithmique de la notion de pathwidth introduite par Robertson et Seymour [RS83]. Kinnersley [Kin92] a prouvé que la pathwidth d'un graphe est égale à sa sommet séparation. Donc l'indice d'échappement sommet, la pathwidth et la sommet séparation sont équivalents : $\mathtt{sn}(G) = \mathtt{pw}(G) + 1 = \mathtt{vs}(G) + 1$. Cependant il n'est pas connu si une équivalence existe avec l'indice de traitement. En d'autres termes, étant donnée la valeur d'un paramètre, est-il possible de calculer l'autre en temps polynomial (la complexité ne dépendant pas de la valeur connue du paramètre) ? Les auteurs de [PHH+00] répondent à cette question lorsque le graphe est un arbre. Rappelons que $\mathtt{vs}(D) \leq \mathtt{pn}(D) \leq \mathtt{vs}(D) + 1$ [CPPS05].

Indice d'échappement arête. Rappelons que pour l'indice d'échappement sommet, le premier joueur peut déplacer le fugitif d'un sommet à un autre si, et seulement si, il existe un chemin dans lequel aucun sommet n'est couvert par un agent. Pour l'indice d'échappement arête, le fugitif peut également se positionner sur une arête du graphe. Ainsi, il y a une troisième action possible pour le second joueur.

(1) placer un agent sur un sommet ;

(2) enlever un agent d'un sommet ;

(3) un agent positionné sur un sommet u peut traverser une arête (u, v).

Comme pour l'indice d'échappement sommet, les sommets et les arêtes peuvent être divisés en trois catégories : les sommets *gardés* sur lesquels des agents sont positionnés, les sommets et arêtes *non sécurisés* sur lesquels le fugitif peut se trouver

et les sommets et arêtes *sécurisés* sur lesquels le fugitif ne peut pas se trouver. Une arête (u, v) devient sécurisée si un agent traverse (u, v) et si le fugitif ne peut pas atteindre (u, v). En d'autres termes, s'il n'existe pas de chemin de la position courante du fugitif vers (u, v) composé uniquement de sommets et arêtes non sécurisés. L'agent est alors positionné sur le sommet v mais il est parfois possible de le retirer juste après. La définition de stratégie monotone est la même que celle donnée dans la Définition 7 : la partie non sécurisée du graphe n'augmente jamais au cours de la stratégie. Rappelons que le second joueur ne connaît pas la position du fugitif.

Définition 8 (p-stratégie, p-strategy) *Une p-stratégie pour un graphe G est une stratégie pour G qui utilise au plus p agents.*

Définition 9 (indice d'échappement arête, edge search number, es) *L'indice d'échappement arête d'un graphe G, noté $\texttt{es}(G)$, est le plus petit p tel qu'il existe une p-stratégie pour G.*

Considérons par exemple un chemin composé des 3 sommets u_1, u_2 et u_3 et des arêtes (u_1, u_2) et (u_2, u_3). Tous les sommets et arêtes sont initialement non sécurisés. Le second joueur peut placer un agent sur u_1. Ensuite il est possible d'utiliser la troisième action afin de parcourir l'arête (u_1, u_2) et positionner l'agent sur u_2. Dans ce cas, le sommet u_1 et l'arête (u_1, u_2) deviennent sécurisés. Dans cet exemple, il est préférable de laisser l'agent sur u_2 et de continuer de manière analogue la stratégie. En effet, enlever l'agent de u_2 rendrait, à nouveau, tout le graphe non sécurisé. Notons que l'indice d'échappement arête de ce graphe est 1 alors que son indice d'échappement sommet est 2.

Dans [KP86], Kirousis *et al.* ont prouvé que $\texttt{sn}(G) - 1 \leq \texttt{es}(G) \leq \texttt{sn}(G) + 1$. Dans [PHH+00], Peng *et al.* ont caractérisé les arbres pour lesquelles il y a égalité : les arbres vérifiant que chacun de ses sommets est incident à une feuille et les arbres sans sommet de degré 2.

Indice de traitement. Nous avons formellement défini l'indice de traitement d'un digraphe D dans la section 2.1.4.3. Pour les digraphes orientés symétriques, il peut être défini comme un jeu des gendarmes et du voleur dans le graphe non orienté sous-jacent. Comme pour l'indice d'échappement sommet, le fugitif est capturé lorsqu'un agent se trouve sur le même sommet mais il peut être également capturé lorsqu'il est entouré d'agents. Cela veut dire que le second joueur (contrôlant les agents) a une troisième action possible :

(1) placer un agent sur un sommet ;

(2) enlever un agent d'un sommet ;

(3) traiter un sommet si tous ses voisins sont couverts par des agents.

Rappelons que le second joueur ne connaît pas la position du fugitif. Un sommet est *traité* si le fugitif ne peut pas être positionné sur ce dernier et le graphe est traité si tous ses sommets sont traités. Nous pouvons (re)définir la notion d'indice de traitement pour un digraphe orienté symétrique en termes de capture de fugitif.

Définition 10 *L'*indice de traitement *d'un graphe G est le nombre minimum d'agents pour capturer le fugitif (pour traiter G).*

Par exemple, une étoile a un indice de traitement de 1, un chemin a un indice de traitement de 2, un cycle a un indice de traitement de 3 et une grille $n \times n$, avec $n \geq 2$, a un indice de traitement de $n+1$.

Coudert *et al.* [CPPS05] ont prouvé que $\texttt{vs}(G) \leq \texttt{pn}(G) \leq \texttt{vs}(G)+1$. Cela implique que $\texttt{sn}(G)-1 \leq \texttt{pn}(G) \leq \texttt{sn}(G)$ et que $\texttt{pw}(G) \leq \texttt{pn}(G) \leq \texttt{pw}(G)+1$.

Nous pouvons, comme pour l'indice d'échappement sommet, diviser les sommets en trois catégories : *gardés*, *non sécurisés* et *sécurisés*.

Définition 11 (p-stratégie de traitement monotone) *Une p-stratégie de traitement est* monotone *si la zone non sécurisée du graphe ne grandit jamais. En d'autres termes, un sommet qui a été traité ne peut pas abriter le fugitif.*

Lemme 1 *Pour tout graphe G, il existe une $\texttt{pn}(G)$-stratégie de traitement monotone pour G.*

Preuve : Soit un graphe $G=(V,E)$. Nous savons que $\texttt{sn}(G)-1 \leq \texttt{pn}(G) \leq \texttt{sn}(G)$ [CPPS05]. Le Lemme 1 est vrai si $\texttt{pn}(G)=\texttt{sn}(G)$ car nous pouvons utiliser le fait qu'il existe une $\texttt{sn}(G)$-stratégie de capture monotone pour G. Si $\texttt{pn}(G)=\texttt{sn}(G)-1$, considérons une $\texttt{pn}(G)$-stratégie de traitement non monotone et soit $X \subseteq V$ l'ensemble des sommets non couverts par des agents au cours de la stratégie. X est un ensemble indépendant de G. Nous construisons alors le graphe $G'=(V',E')$ avec $V'=V \backslash X$ et $E'=E \cup (\cup_{x \in X} \{(u_1,u_2),\ u_1,u_2 \in N(x) \text{ and } u_1 \neq u_2\}) \backslash (\cup_{x \in X} \{(u,x),\ u \in N(x)\})$. En d'autres termes, pour chaque sommet $x \in X$, nous créons une clique entre les voisins $N(x)$ de x, et nous supprimons ensuite x de G ainsi que chacune des arêtes incidentes. À partir de $\texttt{pn}(G)$-stratégie de traitement pour G, nous déduisons une $(\texttt{sn}(G)-1)$-stratégie de capture pour G'. Rappelons que $\texttt{pn}(G)=\texttt{sn}(G)-1$. Si nous considérons uniquement les actions *placer un agent sur un sommet $u \in V$* et *supprimer un agent du sommet $u \in V$* de la stratégie de traitement pour G, alors la séquence d'actions de la stratégie de capture pour G' est exactement la même que la séquence d'actions de la stratégie de traitement pour G. Comme nous avons une $(\texttt{sn}(G)-1)$-stratégie de capture pour G', alors il existe une $(\texttt{sn}(G)-1)$-sratégie de capture monotone pour G' [LaP93]. Enfin, à partir des actions des agents de la $(\texttt{sn}(G)-1)$-stratégie de capture monotone pour G', nous déduisons une $\texttt{pn}(G)$-stratégie de traitement monotone pour G.

En effet, la $\texttt{pn}(G)$-stratégie de traitement monotone pour G est composée de toutes les actions de la $(\texttt{sn}(G)-1)$-stratégie de capture monotone pour G' plus $|X|$ étapes qui consistent à traiter chaque sommet $x \in X$ sans agent. Rappelons que toute stratégie de capture pour une clique couvre simultanément tous les sommets à une certaine étape. Cela veut dire que x peut être traité lorsque tous ses voisins dans G sont couverts par des agents, c'est-à-dire quand tous les sommets de la clique correspondante dans G' (formée par les voisins de x dans G) sont couverts par des agents. Ainsi, pour chaque sommet $x \in X$, nous insérons l'étape qui consiste à traiter x lorsque la précédente condition est vérifiée. □

Corollaire 2 *Si une p-stratégie de traitement pour un graphe G existe, alors il existe une p-stratégie de traitement monotone pour G.*

Pour conclure cette section, une perspective intéressante est de tenter d'établir la relation entre l'indice de traitement d'un digraphe D et un jeu avec des gendarmes et un voleur. Cela permettrait peut être ensuite de lier l'indice de traitement avec la notion de pathwidth dirigée.

2.1.5 Contributions

Le problème de calculer l'indice de traitement étant NP-complet en général, nous décrivons dans la section 2.2 un algorithme distribué permettant de calculer ce paramètre pour tout arbre orienté symétrique. En d'autres termes, notre algorithme permet de déterminer l'ordre sur les requêtes permettant de minimiser le nombre maximum de requêtes simultanément interrompues si le digraphe de dépendances de l'instance correspondante est un arbre orienté symétrique. Notre algorithme peut être executé dans un environnement asynchrone, requiert un temps de calcul de $O(n \log n)$ et l'envoi de n messages de taille $\log_3 n + 4$ bits chacun. Notre contribution principale est la structure de décomposition des arbres que nous proposons (section 2.2.3). En effet, nous montrons dans la section 2.2.5, les possibilités permises par cette structure simple et flexible : mise à jour de manière distribuée de l'indice de traitement après ajout ou suppression de n'importe quelle arête ; calcul de l'indice de traitement d'un arbre où les arêtes sont ajoutées séquentiellement et dans n'importe quel ordre ; extensions de nos algorithmes au calcul des différents paramètres liés à l'indice de traitement décrits précédemment (indice d'échappement sommet, pathwidth, sommet séparation, indice d'échappement arête). Cela est possible en modifiant uniquement les règles d'initialisation. Nos algorithmes sont également valides même si l'arbre est de taille inconnue. Nous donnons enfin quelques perspectives intéressantes comme le calcul de ces paramètres pour d'autres classes de graphes "proches" de l'arbre.

Dans la section 2.3, nous avons étudié le problème d'établir de bons compromis entre le nombre total d'interruptions q et le nombre maximum d'interruptions simultanées p. Nous avons regardé, entre autres, le problème de minimiser p (respectivement q) lorsque q (respectivement p) était fixé. Pour cela, nous avons introduit de nouveaux paramètres visant à calculer cette dégradation. Dans la section 2.3.1, nous montrons la non-approximabilité des différents problèmes visant à calculer ces paramètres de compromis. De plus, nous montrons dans la section 2.3.2 que cette dégradation peut être arbitrairement large. Dans la section 2.3.3, nous étudions la classe des digraphes orientés symétriques. Enfin dans la section 2.3.4, nous expliquons quels sont les autres compromis intéressants à étudier pour le problème de reconfiguration (par exemple la durée d'interruption d'une connexion ou la durée totale de la reconfiguration).

Dans la section 2.4.1, nous montrons comment prendre en compte des requêtes prioritaires qui ne peuvent pas être interrompues durant la reconfiguration. En ef-

fet, nous caractérisons les instances avec requêtes prioritaires admettant une reconfiguration valide des connexions, c'est-à-dire sans interruption de ces requêtes particulières. Nous montrons ensuite que s'il existe une reconfiguration valide alors le problème de reconfiguration avec requêtes prioritaires se ramène au problème de reconfiguration sans requêtes prioritaires. Plus précisément, nous construisons un digraphe D^* à partir du digraphe de dépendances et nous ramenons le problème aux calculs de stratégies de traitement pour D^*. Nous mettons en exergue une borne supérieure sur le nombre d'agents supplémentaires à utiliser. Nous expliquons ensuite deux autres modélisations que nous avons introduites. Dans la section 2.4.2, nous montrons que si toute longueur d'onde peut être partagée par au plus trois connexions, alors le problème de trouver une reconfiguration valide sans interruption est NP-complet. Rappelons que ce problème est très facile dans le modèle initial. Dans la section 2.4.3, nous introduisons un nouveau modèle permettant de prendre en compte certaines contraintes physiques liées à la reconfiguration du routage mais sans se préoccuper des interruptions. Nous montrons par exemple que minimiser le coût d'une reconfiguration est un problème NP-complet même si le réseau est composé de deux nœuds. Nous proposons ensuite des bornes générales et des solutions simples pour certaines instances. Dans [BCM+11], nous proposons également des heuristiques pour ce problème et les comparons via des simulations.

Dans [CHM+09a], nous avons proposé des heuristiques pour le calcul de stratégies de traitement dans les digraphes. Nous sommes actuellement en train de les comparer à deux heuristiques proposées par Jose et Somani [JS03] et par Solano et Pióro [SPa10].

Nous présentons dans la Section 2.5 quelques perspectives concernant le problème de reconfiguration des requêtes de connexion.

2.2 Indice de traitement dans les arbres orientés symétriques

Nous présentons dans cette section un algorithme distribué pour calculer l'indice de traitement dans les arbres orientés symétriques. Des algorithmes centralisés existent pour calculer la pathwidth d'un arbre en temps linéaire [EST94, Sch90, Sko03, Gol91], mais pas d'algorithme distribué.

Notre Algorithme, `Algorithme MHD`, étend l'algorithme proposé par Ellis *et al.* [EST94] pour l'indice d'échappement sommet. Il peut être executé dans un environnement asynchrone, requiert un temps de calcul de $O(n \log n)$ et n messages de taille $\log_3 n + 4$ bits chacun. La contribution principale réside dans la structure de données proposée dans l'`Algorithme MHD`, appelée *décomposition hierarchique*. Cette structure simple et flexible permet plusieurs opérations : mise à jour distribuée de l'indice de traitement après ajout ou suppression de n'importe quelle arête ; calcul de l'indice de traitement dans un arbre où les arêtes sont ajoutées séquentiellement et dans n'importe quel ordre (`Algorithme incrémental MHD`) ; calcul des invariants de graphes présentés précédemment (indice d'échappement sommet, pa-

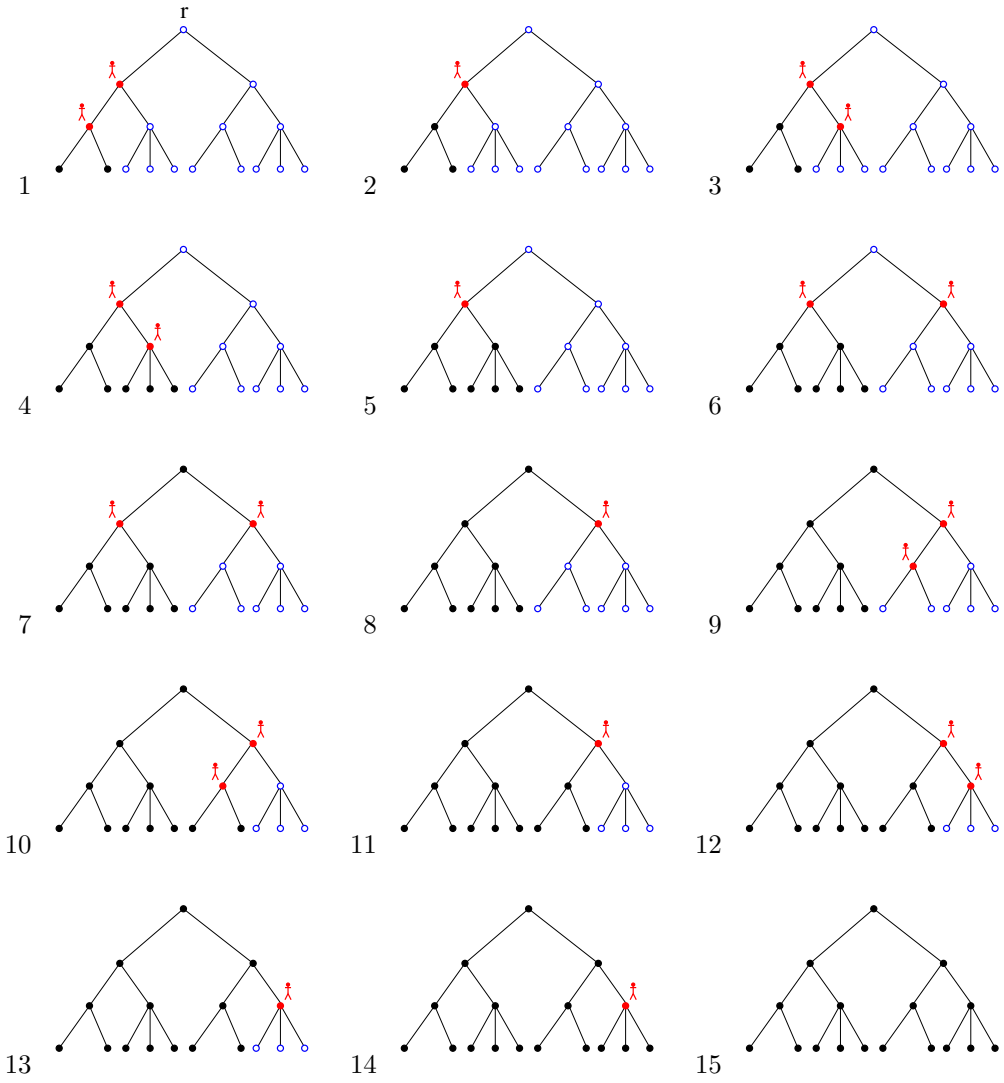


FIG. 2.5 – 2-stratégie de traitement monotone.

thwidth, sommet séparation, indice d'échappement arête) en modifiant uniquement les règles d'initialisation ; extension d'`Algorithme MHD` pour les arbres et forêts de tailles inconnues (messages de tailles au plus $2\log_3 n + 5$ bits).

Dans notre article paru dans Algorithmica [CHM11], nous présentons l'`Algorithme MHD` et l'`Algorithme incrémental MHD` pour l'indice d'échappement sommet avant de l'étendre aux autres invariants (comme l'indice de traitement notamment). Nous avons choisi ici, par un souci de cohérence, de présenter l'`Algorithme MHD` et l'`Algorithme incrémental MHD` pour l'indice de traitement.

2.2.1 Préliminaires et définitions

Nous commençons par rappeler un théorème central relatif à la valeur de tous les paramètres que nous calculons dans les arbres.

Théorème 8 ([Par78, PHH+00, CPPS05, CS11]) *Étant donnés un arbre T et un entier $p \geq 1$, $\texttt{es}(T) \geq p+1$ si et seulement si T a un sommet v pour lequel il y a au moins trois branches T_i, $i = 1, 2, 3$, telles que $\texttt{es}(T_i) \geq p$. Cela est également vrai pour $\texttt{vs}(T)$ et $\texttt{pw}(T)$, mais aussi pour $\texttt{sn}(T)$ et $\texttt{pn}(T)$ quand $p \geq 2$.*

Le Théorème 8 a été initialement prouvé par Parson pour l'indice d'échappement arête [Par78]. Plus tard, la preuve a été adaptée pour l'indice d'échappement sommet [PHH+00] et pour l'indice de traitement [CPPS05, CS11] (quand $p \geq 2$). Le théorème est également vrai pour $\texttt{pw}(T)$ et $\texttt{vs}(T)$. Nous avons ainsi une construction forçant l'indice d'échappement arête à augmenter de 1. Le sommet v du Théorème 8 est communément appelé un Parsons node [Par78]. Enfin, le Théorème 8 implique que pour tout arbre T ; $\texttt{sn}(T)$, $\texttt{pw}(T)$, $\texttt{vs}(T)$, $\texttt{pn}(T)$ et $\texttt{es}(T)$ sont au plus $\log_3(n)$, avec n le nombre de sommets de T. Cela peut être prouvé par induction sur la valeur du paramètre. En effet, la taille minimum d'un arbre de paramètre égal à $p+1$ est au moins trois fois la taille minimum d'un arbre de paramètre égal à p.

Nous introduisons dans les deux prochaines définitions des stratégies de traitement particulières qui nous serviront dans la conception d'`Algorithme MHD` et d'`Algorithme incrémental MHD`.

Définition 12 (p-stratégie de traitement terminant en v) *Étant donnés un graphe $G = (V, E)$ et un sommet $v \in V$, une p-stratégie de traitement* termine en v *si la dernière action est supprimer l'agent du sommet v.*

Définition 13 (p-stratégie de traitement débutant en v) *Étant donnés un graphe $G = (V, E)$ et un sommet $v \in V$, une p-stratégie de traitement* débute en v *si v est le premier sommet couvert par un agent.*

La figure 2.5 décrit une 2-stratégie de traitement monotone pour un arbre de 17 sommets. Il est assez facile de remarquer qu'il n'existe pas de 2-stratégie de traitement terminant (ou débutant) avec un agent sur la racine de l'arbre r (figure 2.5(a)).

Remarque 1 *Comme nous considérons seulement des stratégies de traitement monotones, une p-stratégie de traitement débutant en v assure que durant toutes les prochaines étapes, le fugitif ne peut pas se trouver sur v.*

Le Lemme 2 montre que les deux notions *débuter* et *finir* la stratégie de traitement sur un sommet donné, sont équivalentes pour les stratégies de traitement monotones.

Lemme 2 *Étant donnés un graphe $G = (V, E)$ et un sommet $v \in V$, il existe une p-stratégie de traitement monotone débutant en v pour G si, et seulement si, il existe une p-stratégie de traitement monotone finissant en v pour G.*

Preuve : Soit S une séquence de $x \leq 2|V|$ actions $m_1, m_2, \ldots, m_x$ décrivant une p-stratégie de traitement monotone pour G finissant en v, c'est-à-dire finissant avec l'action m_x *retirer un agent d'un sommet* v. Nous notons V^{cov} l'ensemble des sommets couverts par un agent au cours de la p-stratégie de traitement défini par la séquence S et nous notons $\overline{V^{cov}}$ l'ensemble des sommets non couverts.

Soit $\overline{m}_i$ l'action *placer un agent sur un sommet* $u \in V^{cov}$ (*retirer un agent d'un sommet* $u \in V^{cov}$, respectivement) si m_i est l'action *retirer un agent d'un sommet* $u \in V^{cov}$ (*placer un agent sur un sommet* $u \in V^{cov}$, respectivement). De plus, soit $\overline{m}_i$ l'action *traiter un sommet* $u \in \overline{V^{cov}}$ si m_i est l'action *traiter un sommet* $u \in \overline{V^{cov}}$. Soit $\overline{S}$ la séquence d'actions $\overline{m}_{\sigma(1)}, \overline{m}_{\sigma(2)}, \ldots, \overline{m}_{\sigma(x)}$ avec $\sigma(i) = x - i + 1$.

Nous prouvons maintenant que $\overline{S}$ donne une p-stratégie de traitement monotone débutant en v, c'est-à-dire débutant avec l'action $\overline{m}_x =$ *'placer un agent sur le sommet* v*'*. Notons premièrement que si $\overline{S}$ donne une stratégie de traitement débutant en v, alors nécessairement celle-ci est monotone. En effet, dans la stratégie ainsi obtenue, chaque sommet est traité une unique fois. De plus si $\overline{S}$ est une stratégie de traitement, alors cette dernière est une p-stratégie de traitement. En effet si nous numérotons les p agents, alors tout sommet $u \in V^{cov}$ couvert par l'agent i, $1 \leq i \leq p$, dans S, est aussi couvert par l'agent i dans $\overline{S}$.

Supposons que $\overline{S}$ n'est pas une stratégie de traitement. Il y a quatre cas possibles.

A) Dans $\overline{S}$, il y a une arête $(u, v) \in E$ telle que *1)* nous plaçons un agent sur u, *2)* nous retirons ensuite cet agent de u, *3)* nous plaçons ensuite un agent sur v, *4)* et enfin nous retirons l'agent de v.

Notons que ces étapes ne sont pas nécessairement consécutives et qu'il n'y a pas d'autres actions concernant u ou v dans $\overline{S}$ car S est une stratégie de traitement monotone valide. En conséquence, cela veut dire que, dans S, *1)* nous plaçons un agent sur v, *2)* nous retirons ensuite cet agent de v, *3)* nous plaçons ensuite un agent sur u, *4)* et enfin nous retirons l'agent de u. Une contradiction car S est une p-stratégie de traitement valide.

B) Dans $\overline{S}$, il y a une arête $(u, v) \in E$ telle que *1)* nous plaçons un agent sur u, *2)* nous retirons ensuite l'agent de u, *3)* et enfin nous traitons le sommet v.

Dans la stratégie de traitement définie par S, ces actions correspondent à *1)* traiter v, *2)* placer un agent sur u, *3)* et retirer l'agent de u. Cela n'est pas une stratégie de traitement valide car nous traitons v alors qu'un de ses voisins, u, n'est ni traité ni occupé par un agent. Une contradiction car S est une p-stratégie de traitement valide.

C) Dans $\overline{S}$, il y a une arête $(u, v) \in E$ telle que *1)* nous traitons v, *2)* nous plaçons ensuite un agent sur u, *3)* et nous retirons enfin l'agent de u.

Dans S, cela correspond à *1)* placer un agent sur u, *2)* retirer l'agent de u, *3)* et traiter v. Une contradiction car S est une p-stratégie de traitement valide.

D) Dans $\overline{S}$, il y a une arête $(u, v) \in E$ telle que *1)* nous traitons v, *2)* et nous traitons u.

Dans S, cela correspond à *1)* traiter u, *2)* et traiter v. Une contradiction car S est une p-stratégie de traitement valide. □

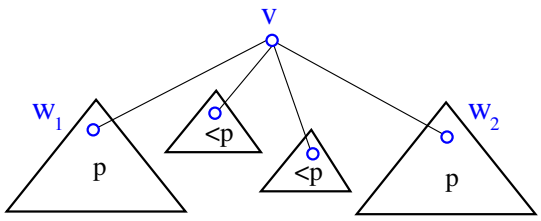


FIG. 2.6 – Arbre instable générique T_v enraciné en v, avec $\texttt{pn}(T_v) = p > 1$. Il y a deux sous-arbres stables T_{w_1} et T_{w_2} enracinés en w_1 et w_2 (deux fils de v) respectivement, avec $\texttt{pn}(T_{w_1}) = \texttt{pn}(T_{w_2}) = p$. Les autres sous-arbres ont des indices de traitement au plus $p - 1$.

Remarquons qu'étant donné un graphe $G = (V, E)$, pour tout sommet $v \in V$, une p-stratégie de traitement peut être transformée en une $(\leq p + 1)$-stratégie de traitement terminant (ou débutant) en v en ajoutant, si nécessaire, un $(p+1)^e$ agent en v et le laissant durant toute la p-stratégie de traitement.

2.2.2 Arbre stable et arbre instable

Nous définissons deux catégories d'arbres qui jouent un rôle très important dans l'`Algorithme MHD` et nous décrivons leurs propriétés qui nous seront utiles pour la suite.

Définition 14 (arbre stable) *Un arbre T_v enraciné en le sommet v, avec $\texttt{pn}(T_v) = p$, est un* arbre stable *s'il existe une p-stratégie de traitement pour T_v qui termine (débute) en v.*

Définition 15 (arbre instable) *Un arbre T_v enraciné en le sommet v, avec $\texttt{pn}(T_v) = p > 1$, est un* arbre instable *s'il existe deux sous-arbres stables T_{w_1} et T_{w_2} (enracinés en deux fils de v : w_1 et w_2, respectivement) tels que $\texttt{pn}(T_{w_1}) = \texttt{pn}(T_{w_2}) = p$ et tous les autres sous-arbres $T_{w_3}, \ldots, T_{w_j}$, enracinés en chacun des autres fils de v : $w_3, \ldots, w_j$, respectivement, sont tels que $\texttt{pn}(T_{w_i}) < p$, $3 \leq i \leq j$, (la figure 2.6 montre un arbre instable générique).*

L'arbre T_r enraciné en r de la figure 2.5 n'est pas stable car il exsite une 2-stratégie de traitement monotone mais il n'existe pas de 2-stratégie de traitement monotone débutant (terminant) en r. Nous pouvons remarquer que T_r est instable car les deux sous-arbres enracinés en les deux voisins de r sont stables et d'indice de traitement 2 (Définition 15).

Remarque 2 *Un arbre T_v enraciné en v peut être ni stable ni instable.*

Propriété 1 *Étant donné un arbre instable T_v enraciné en v, avec $\texttt{pn}(T_v) = p > 1$, il n'existe pas de p-stratégie de traitement terminant (débutant) en v.*

Preuve : Soit T_v un arbre instable enraciné en v. Soient $w_1, \ldots, w_j$ les fils de v et soient T_{w_1} et T_{w_2} les deux sous-arbres instables tels que $\texttt{pn}(T_{w_1}) = \texttt{pn}(T_{w_2}) = p$. L'arbre T_v est représenté dans la figure 2.6. Supposons qu'il existe une p-stratégie de traitement pour T_v débutant en v. Cela donne naturellement une p-stratégie de traitement pour $T_{w_1} \cup T_{w_2} \cup \{v\}$ débutant en v. Une telle stratégie de traitement commence par traiter un des deux sous-arbres T_{w_1} et T_{w_2}, disons T_{w_1}. Mais tant que la p-stratégie de traitement est en train de traiter T_{w_1}, par définition d'une p-stratégie de traitement débutant en v, il est garanti que le fugitif ne peut pas aller sur v, sinon v serait recontaminé (*i.e. le fugitif peut aller sur v, alors qu'un agent a déjà été placé sur v*) ce qui violerait l'hypothèse que nous considérons uniquement des p-stratégies de traitement monotones. Ainsi, un agent doit être placé sur v ou sur certains sommets de T_{w_2}. Donc une stratégie de traitement débutant en v nécessite au moins $p+1$ agents. Rappelons que nous considérons uniquement des stratégies de traitement qui ne placent jamais strictement plus qu'un agent par sommet.

Par le Lemme 2, nous obtenons le même résultat pour une p-stratégie de traitement finissant en v. □

Remarque 3 *Un arbre T_v enraciné en v avec $\texttt{pn}(T_v) = 1$ n'est jamais considéré comme instable même s'il est nécessaire d'utiliser 2 agents pour terminer la stratégie de traitement en v.*

Définition 16 *Un arbre T_v enraciné en le sommet v, avec $\texttt{pn}(T_v) = 1$, est un $(1,2)$-arbre si et seulement si le plus petit p tel qu'une p-stratégie de traitement débutant (ou finissant) en v existe, est $p = 2$.*

Lemme 3 *Étant donnés un arbre $T = (V, E)$ et un sous-arbre instable T_v enraciné en $v \in V$, avec $\texttt{pn}(T_v) = p > 1$, alors $\texttt{pn}(T) = p$ si et seulement si $\texttt{pn}(T \setminus T_v) \leq p-1$.*

Preuve : Soient T un arbre et T_v un sous-arbre instable enraciné en v. Soient T_{w_1} et T_{w_2} deux arbres stables (enracinés en deux fils de v : w_1 et w_2, respectivement) tels que $\texttt{pn}(T_{w_1}) = \texttt{pn}(T_{w_2}) = p$. La figure 2.6 représente l'arbre T_v.

Si $\texttt{pn}(T \setminus T_v) \geq p$, alors il y a trois arbres disjoints enracinés en trois différents voisins de v, chacun ayant un indice de traitement au moins p. Si un d'entre eux a un indice de traitement strictement plus grand que p, alors $\texttt{pn}(T) \geq p+1$. Sinon, les trois sous-arbres ont un indice de traitement de p et par le Théorème 8, $\texttt{pn}(T) = p+1$.

Sinon $\texttt{pn}(T \setminus T_v) \leq p-1$ et nous décrivons une p-stratégie de traitement pour T. Nous débutons par une p-stratégie de traitement pour T_{w_1} finissant avec un agent sur w_1. Nous plaçons ensuite un agent sur v et retirons celui sur w_1. Nous continuons avec une $(\leq p-1)$-stratégie de traitement pour $T_v \setminus (T_{w_1} \cup T_{w_2} \cup \{v\})$. Comme $\texttt{pn}(T \setminus T_v) \leq p-1$, nous continuons avec une $(\leq p-1)$-stratégie de traitement pour $T \setminus T_v$. Nous plaçons ensuite un agent sur w_2 et retirons celui sur v. Nous finissons avec une p-stratégie pour T_{w_2} débutant en w_2 (cette stratégie existe par hypothèse). □

Étant donné un arbre $T = (V, E)$, s'il existe un sous-arbre instable T_u enraciné en $u \in V$, avec $\texttt{pn}(T_u) = p$, par le Lemme 3, calculer $\texttt{pn}(T \setminus T_u)$ nous permet de

décider si $\texttt{pn}(T) = p$ ou non. En calculant $\texttt{pn}(T \setminus T_u)$, si aucun autre sous-arbre instable n'est trouvé, la valeur exacte de $\texttt{pn}(T)$ peut être calculée. Mais si un autre sous-arbre instable $T_{u'}$ est trouvé, nous devons alors résoudre le même problème de décision. Ainsi nous devons calculer $\texttt{pn}(T \setminus (T_u \cup T_{u'}))$, et ainsi de suite. La figure 2.7 représente ce problème récursivement.

Considérons l'arbre T_v enraciné en v de la figure 2.7. Il est composé de 6 sous-arbres disjoints : T^1, T^2, T^3, T^4, T^5 sont instables alors que T^0 enraciné en v est stable, avec indice de traitement 4, 5, 6, 7, 8 et 3, respectivement. T^5 est le sous-arbre instable de plus grand indice de traitement ($\texttt{pn}(T^5) = 8$) et par le Lemme 3, nous savons que $\texttt{pn}(T_v) = 8$ si et seulement si $\texttt{pn}(T_v \setminus T^5) \leq 7$. Alors considérons $T_v \setminus T^5$. T^4 est le sous-arbre instable de plus grand indice de traitement ($\texttt{pn}(T^4) = 7$) et par le Lemme 3, nous savons que $\texttt{pn}(T_v \setminus T^5) = 7$ si et seulement si $\texttt{pn}(T_v \setminus (T^5 \cup T^4)) \leq 6$. Et ainsi de suite. Finalement, nous obtenons que $\texttt{pn}(T^0) \leq 3$ et donc que $\texttt{pn}(T_v) = 8$.

Pour formaliser les remarques précédentes, nous introduisons la notion centrale de l'`Algorithme MHD` dans la section 2.2.3.

2.2.3 Décomposition hiérarchique (minimale)

La clé d'`Algorithme MHD` est la représentation de tout arbre par ce que nous appelons une *décomposition hiérarchique* (Définition 17) et une *décomposition hiérarchique minimale* (Définition 18). En effet, étant donné un arbre T, l'`Algorithme MHD` calcule une décomposition hiérarchique minimale de T avant de déduire son indice de traitement.

Définition 17 (décomposition hiérarchique) *Étant donné un arbre T_r enraciné en r, une* décomposition hiérarchique *de T_r, notée $HD(T_r)$, est une famille $\{T^i\}_{0 \leq i \leq k}$ d'arbres telle que :*

- *l'ensemble des sous-arbres $\{T^i\}_{0 \leq i \leq k}$ de T_r induit une partition des sommets de T_r (i.e., les sous-arbres sont disjoints et couvrent tous les sommets) ;*
- *T^0 est soit stable, soit un $(1,2)$-arbre, soit instable et est enraciné en $v_0 = r$;*
- *T^i est instable et est enraciné en v_i, $1 \leq i \leq k$;*
- *si deux arbres T^i et T^j, $0 \leq i \leq k$, $0 \leq j \leq k$, $i \neq j$, sont tels que le chemin allant de v_i à r dans T_r passe par v_j, alors $\texttt{pn}(T^i) > \texttt{pn}(T^j)$.*

Nous associons à $HD(T_r)$ la paire $((p,p'), vect)$ avec $p = p'$ si T^0 est un arbre stable ($p \geq 0$), $p = 1$ et $p' = 2$ si T^0 est un $(1,2)$-arbre et $p = p' = -1$ si T^0 est un arbre instable, et vect un vecteur de taille $L(vect)$ avec $L(vect)$ le plus grand indice de traitement parmi les arbres instables de $HD(T_r)$. vect contient dans la cellule i, notée $vect[i]$, le nombre d'arbres instables de $HD(T_r)$ d'indice de traitement i.

La décomposition hiérarchique $HD(T_v)$ de l'arbre T_v enraciné en v de la figure 2.7, est représentée par la paire $((p,p'), vect)$ de la table 2.1. La paire représentant la décomposition hiérarchique du sous-arbre $T_{v'}$ y est également décrite.

Remarque 4 *Étant donnée une paire $((p,p'), vect)$ associée à une décomposition hiérarchique, nous avons $vect[1] = 0$. La remarque 3 assure cela.*

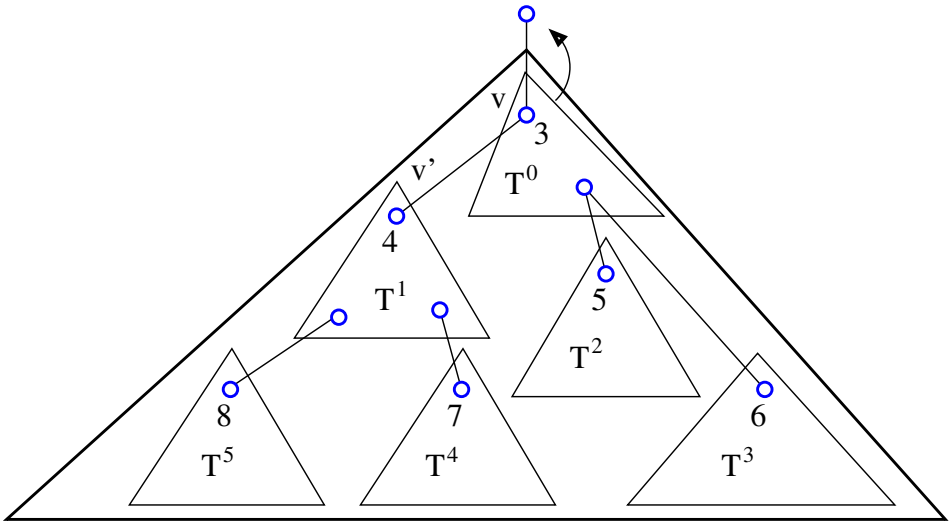


FIG. 2.7 – La décomposition hiérarchique $MHD(T_v)$ de l'arbre T_v enraciné en v contient six arbres disjoints : T^1, T^2, T^3, T^4, T^5 sont instables alors que T^0 enraciné en v est stable. Les indices de traitement de ces arbres sont indiqués par les entiers sur les racines. Les arbres disjoints respectent la contrainte sur l'ordre. Par exemple, il y a une arête entre la racine de T^4 et un sommet de T^1.

Remarque 5 *Une décomposition hiérarchique d'un arbre T_r enraciné en r est associée à une unique paire $((p,p'),vect)$ mais une paire $((p,p'),vect)$ peut être associée à plusieurs décompositions hiérarchiques. En effet, les arêtes connectant les différents arbres de la partition de la décomposition hiérarchique n'ont pas d'influence sur la paire $((p,p'),vect)$ tant que les propriétés de la décomposition hiérarchique sont respectées. Considérons par exemple l'arbre T_v de la figure 2.7. L'arbre T^5, avec $\mathtt{pn}(T^5) = 8$, peut être attaché à T^3 ayant $\mathtt{pn}(T^3) = 6$ (au lieu de T^1) sans modifier la représentation de la décomposition hiérarchique de T_v décrite dans la table 2.1.*

	(p,p')	$vect[1]$	$vect[2]$	$vect[3]$	$vect[4]$	$vect[5]$	$vect[6]$	$vect[7]$	$vect[8]$
$HD(T_v) = MHD(T_v)$	(3,3)	0	0	0	1	1	1	1	1
$HD(T_{v'}) = MHD(T_{v'})$	$(-1,-1)$	0	0	0	1	0	0	1	1

TAB. 2.1 – Paires $((p,p'),vect)$ associées aux décompositions hiérarchiques $HD(T_v)$ et $HD(T_{v'})$ des arbres T_v et $T_{v'}$ de la figure 2.7. Dans cet exemple, les décompositions hiérarchiques sont aussi les décompositions hiérarchiques minimales $MHD(T_v)$ et $MHD(T_{v'})$.

Après avoir défini la notion de décomposition hiérarchique, nous introduisons ici une décomposition hiérarchique particulière.

Définition 18 (décomposition hiérarchique minimale) *Étant donné un arbre T_r enraciné en r, une* décomposition hiérarchique minimale *de T_r, notée $MHD(T_r)$, est une décomposition hiérarchique de T_r (Définition 17) telle que $\forall i, j \in [0, k]$, $i \neq j$, nous avons $\texttt{pn}(T^i) \neq \texttt{pn}(T^j)$.*

L'existence d'une décomposition hiérarchique minimale est assurée par le Théorème 9 (prouvé dans la section 2.2.4). Plus précisément, le Théorème 9 implique qu'il existe un sommet v pour lequel une décomposition hiérarchique minimale de T_v existe. `Algorithme MHD` peut être légèrement modifié pour obtenir une décomposition hiérarchique minimale de T enraciné en n'importe quel sommet r. Il suffit de forcer r à ne pas envoyer de message.

Propriété 2 *Étant donnée la représentation $((p, p'), vect)$ de la décomposition hiérarchique minimale $MHD(T_r)$ d'un arbre T_r enraciné en r, $\forall i \in [2 \dots L(vect)]$, nous avons $vect[i] \in \{0, 1\}$ et $vect[1] = 0$.*

La Propriété 2 se déduit directement de la Définition 18 et de la remarque 4. Avant de présenter `Algorithme MHD` dans la section 2.2.4, nous prouvons le Lemme 4 qui permet de calculer l'indice de traitement de tout arbre à partir de sa décomposition hiérarchique minimale.

Lemme 4 *Étant donnée la représentation $((p, p'), vect)$ de la décomposition hiérarchique minimale $MHD(T_r)$ de l'arbre T_r enraciné en r, nous avons $\texttt{pn}(T_r) = \max(p, L(vect))$.*

Preuve : Rappelons que $L(vect)$ est le plus grand indice de traitement parmi les arbres instables de $HD(T_r)$. Si $p \geq L(vect)$, alors $MHD(T_r)$ est composé d'un unique arbre stable $T^0 = T_r$ et $\texttt{pn}(T_r) = p$.

Si $p < L(vect)$, alors nous prouvons l'assertion par induction sur $L(vect)$. Comme $MHD(T_r)$ est une décomposition hiérarchique minimale de T_r, il y a un unique arbre instable T^k tel que $\texttt{pn}(T^k) = L(vect)$. Donc en considérant $MHD(T_r)$ privé de l'arbre T^k, nous obtenons une décomposition hiérarchique minimale $MHD(T_r \setminus T^k)$ de $T_r \setminus T^k$. Ainsi, la taille du vecteur associé à $MHD(T_r \setminus T^k)$ est strictement inférieure à $L(vect)$. Par hypothèse d'induction, nous avons $\texttt{pn}(T_r \setminus T^k) < L(vect)$.

Décrivons une $L(vect)$-stratégie de traitement pour T_r. Nous débutons par une $L(vect)$-stratégie de traitement pour T^k. Il en existe une dans laquelle, à une certaine étape, un agent est positionné sur la racine v_k et aucun autre agent n'occupe un autre sommet. À cette étape, nous insérons une $(\leq L(vect) - 1)$-stratégie de traitement pour $T_r \setminus T^k$. Nous finissons ensuite la $L(vect)$-stratégie de traitement pour T^k. □

2.2.4 Algorithme distribué pour l'indice de traitement

l'Algorithme 1 décrit l'`Algorithme MHD` qui construit la décomposition hiérarchique minimale $MHD(T_r)$ et calcule l'indice de traitement $\texttt{pn}(T)$ de tout arbre $T = (V, E)$ avec $r \in V$.

Algorithme 1 : `Algorithme MHD`

- Chaque feuille envoie le message d'initialisation $((0,0),[\,])$, avec $[\,]$ représentant une vecteur de taille 0, à son unique voisin qui devient son parent.
- Un sommet $v \in V$ qui a reçu des messages de tous ses voisins sauf un, calcule la paire $((p,p'),vect)$ représentant la décomposition hiérarchique minimale $MHD(T_v)$ de T_v en utilisant l'Algorithme 2. Le sommet v envoie ensuite le message $((p,p'),vect)$ à son dernier voisin qui devient son parent.
- Le dernier sommet $r \in V$ est la racine de T. Lorsque ce dernier a reçu un message de tous ses voisins, il calcule la paire $((p,p'),vect)$ représentant la décomposition hiérarchique minimale $MHD(T_r)$ de $T_r = T$ en utilisant l'Algorithme 2. Le Lemme 4 nous donne ensuite $\mathtt{pn}(T)$. Il est possible que deux sommets reçoivent un message de tous leurs voisins respectifs. La Remarque 6 permet de résoudre ce problème.

Remarque 6 *Il est possible que deux sommet adjacents v et w reçoivent un message de tous leurs voisins. Cela arrive lorsque v, après avoir envoyé son message à son dernier voisin w, reçoive également un message de w. Dans ce cas, v et w sont candidats pour devenir la racine de l'arbre. Pour pallier ce problème, le plus simple est certainement de choisir le sommet de plus grand identifiant (en supposant un ordre total sur les identifiants des sommets comme l'adresse MAC par exemple). Cela évite la transmission de bits supplémentaires comme nous pouvons supposer que chaque sommet connaît l'identifiant de tous ses voisins. Sinon, nous pouvons utiliser un mécanisme classique d'élection de leader pour déterminer la racine [Pel90]. Cela peut être réalisé avec* $\log(n)$ *bits.*

Lemme 5 *Soit T_v^0 un arbre enraciné en le sommet v tel que les fils de v sont enracinés à des (éventuellement vides) arbres stables ou $(1,2)$-arbres de paires $(p_1,p'_1),\ldots,(p_{d-1},p'_{d-1})$. L'Algorithme 3 calcule la paire (q,q') associée à l'arbre T_v^0 avec $q = \mathtt{pn}(T_v^0)$ et si T_v^0 est un arbre stable, alors $q' = q$, sinon $q' = q+1$ (arbre instable ou $(1,2)$-arbre).*

Preuve : Un sommet $v \in V$, recevant les paires $(p_1,p'_1),\ldots,(p_{d-1},p'_{d-1})$ de ses voisins $v_1,\ldots,v_{d-1}$, avec $\forall i, p_i < 2$, calcule la paire (q,q'). Nous prouvons les cas d'initialisation (ligne 3 et ligne 4).

- Si v est une feuille, v ne reçoit aucun message, et donc $int_{max} = -1$. Alors l'Algorithme 3 retourne $(q,q') = (0,0)$. Cela est correct car l'indice de traitement d'un sommet est 0.
- Si tous les voisins envoyant un message sont des feuilles, alors $int_{max} = 0$ et l'Algorithme 3 retourne $(q,q') = (1,1)$. L'indice de traitement d'une étoile (de centre v) est 1.
- Si v reçoit un unique message d'un sommet qui est le centre d'une étoile, alors $|I| = 1$ et $(p_i,p'_i) = (1,1), i \in I$. L'Algorithme 3 retourne $(q,q') = (1,2)$. Cela est correct car l'indice de traitement d'une étoile est 1, mais une stratégie de traitement finissant (débutant) en v nécessite 2 agents.

Algorithme 2 : calcul de la représentation $((p,p'),vect)$ de la décomposition hiérarchique minimale $MHD(T_v)$ de l'arbre T_v enraciné en v

Précondition : représentations $((p_1,p'_1),vect_1),\ldots,((p_{d-1},p'_{d-1}),vect_{d-1})$ des décompositions hiérarchiques minimales $MHD(T_{v_1}),\ldots,MHD(T_{v_{d-1}})$ des sous-arbres $T_{v_1},\ldots,T_{v_{d-1}}$ enracinés en les fils de v : $v_1,\ldots,v_{d-1}$.

Précondition : un vecteur $vect_{sum}$ tel que $vect_{sum}[i] := vect_1[i]+\ldots+vect_{d-1}[i]$, $\forall i \in [2, \max_{1\leq j\leq d-1} L(vect_j)]$ et $vect_{sum}[1]=0$.

Postcondition : $((p,p'),vect)$ est la représentation de la décomposition hiérarchique minimale $MHD(T_v)$ de T_v.

1: Soit (q,q') la paire calculée par l'Algorithme 3 à partir des valeurs des arbres stables (éventuellement vides) de $MHD(T_{v_1}),\ldots,MHD(T_{v_{d-1}})$: $(p_1,p'_1),\ldots,(p_{d-1},p'_{d-1})$.
2: **si** $1<q<q'$ **alors** %/* *L'union des arbres stables est instable* */
3: $L(vect) := \max(L(vect_{sum}),q)$; $vect[j]:=0, \forall j\in[1,L(vect)]$
4: $vect := vect_{sum}$
5: $vect[q] := vect[q]+1$
6: $vect[j]:=0, \forall j\in[2,p-1]$
7: $(p,p') := (-1,-1)$
8: **sinon** %/* $q == q'$ *et donc l'union des arbres stables est stable ou* $(q,q')==(1,2)$ */
9: $L(vect) := L(vect_{sum})$
10: $vect := vect_{sum}$
11: $vect[j]:=0, \forall j\in[2,q-1]$
12: Soit k tel que $vect[k]>1$ et $vect[i]\leq 1, \forall i\in[k+1,L(vect)]$ /* si k n'existe pas, alors $k:=-1$ */
13: Soit $k_1 > \max(k,q-1)$ tel que $vect[k_1]=0$ and $vect[i]\geq 1, \forall i\in[\max(k,q),k_1-1]$
/* nous supposons qu'il existe une cellule virtuelle $vect[L(vect)+1]=0$ */
/* si k_1 n'existe pas, alors $k_1:=-1$ */
14: **si** $k_1>1$ **alors**
15: $vect[i]:=0, \forall i\in[2,k_1]$
16: $(p,p') := (k_1,k_1)$
17: **sinon** %/* *si* $k_1 == -1$, *alors la décomposition hiérarchique est minimale* */
18: $(p,p') := (q,q')$
19: **si** $vect[i]==0, \forall i\leq L(vect)$ **alors**
20: $vect := [\,]$ /* [ ] est un vecteur de taille 0 */
21: retourner $((p,p'),vect)$

Algorithme 3 : calcul de (q, q')

Précondition : une liste de paires d'entiers $(p_1, p'_1), \ldots, (p_{d-1}, p'_{d-1})$: étant donné un arbre T_v^0 enraciné en v avec les fils de v racines des arbres stables (éventuellement vides) ou racines de $(1,2)$-arbres correspondants aux paires $(p_1, p'_1), \ldots, (p_{d-1}, p'_{d-1})$.

Postcondition : $q = \mathtt{sn}(T_v^0)$ et q' est la valeur minimum telle que une q'-stratégie de traitement finissant (ou débutant) avec un agent sur v existe pour T_v^0. Rappelons que $q \leq q' \leq q+1$.

1: $int_{max} := \max_{1 \leq j \leq d-1}\{int_j\}$ /* $int_{max} := -1$ `s'il n'y a pas d'arbre stable` */
2: $I := \{i; int_i = int_{max}\}$ /* `tous les` i `tels que` int_i `est maximum` */
3: **si** $int_{max} < 2$ **alors**
4: $$(q, q') := \begin{cases} (0,0) & \text{when } int_{max} = -1 \\ (1,1) & \text{when } int_{max} = 0 \\ (1,2) & \text{when } |I| = 1 \text{ and } (p_i, p'_i) = (1,1), i \in I \\ (2,2) & \text{otherwise} \end{cases}$$
5: **sinon** %/* *cas généraux* */
6: **si** $|I| == 2$ **alors** %/* T_v^0 *est instable* */
7: $(q, q') := (int_{max}, int_{max} + 1)$
8: **sinon** %/* T_v^0 *est stable* */
9: **si** $|I| > 2$ **alors** %/* *Théorème 8* */
10: $(q, q') := (int_{max} + 1, int_{max} + 1)$
11: **sinon** %/* $I = 1$ */
12: $(q, q') := (int_{max}, int_{max})$
13: retourner (q, q')

– Si v reçoit un message d'un sommet qui est dans une étoile (mais pas le centre) et des messages de feuilles (d'indice de traitement 0) : $|I| = 1$ et $(p_i, p'_i) = (1,2), i \in I$; ou si v reçoit des messages d'au moins deux sommets d'indice de traitement 1 et de sommets feuilles : $|I| \geq 2$ et $p_i = 1, i \in I$. Dans les deux cas, l'Algorithme 3 retourne $(q, q') = (2,2)$. Cela est correct car 2 agents sont nécessaires et suffisants pour terminer (débuter) la stratégie de traitement en v dans ces situations.

La ligne 6 et la ligne 7 traitent le cas où exactement deux arbres stables ont l'indice de traitement maximum $p > 1$. Dans ce cas, l'arbre T_v^0 est un arbre instable avec $\mathtt{pn}(T_v^0) = p$ (Définition 15 de la section 2.2.2). Donc l'Algorithme 3 retourne $(q, q') := (p, p+1)$.

La ligne 9 et la ligne 10 représentent le cas où il y a strictement plus que deux arbres instable d'indice de traitement maximum $p > 1$. Dans ce cas, le Théorème 8 montre que $\mathtt{pn}(T_v^0) = p+1$. De plus, il existe une $(p+1)$-stratégie de traitement finissant en v : nous plaçons un agent sur v, nous continuons avec des $(\leq p)$-stratégies de traitement pour les arbres stables enracinés en les fils de v (de manière séquentielle), et enfin nous traitons v en retirant l'agent positionné. Donc l'Algorithme 3 retourne $(q, q') := (p+1, p+1)$.

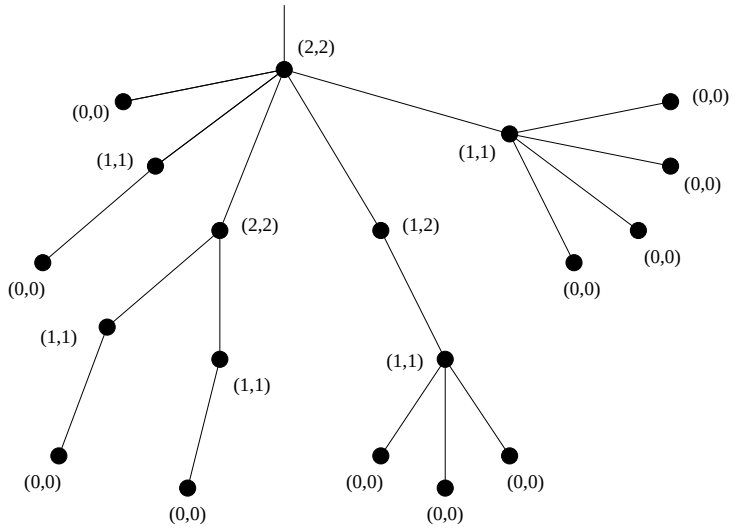


FIG. 2.8 – Exemples de différents cas d'initialisation pour le calcul de l'indice de traitement. Les paires sur les sommets représentent les entiers (p, p') correspondant aux différents sous-arbres.

Enfin, s'il y a un unique arbre stable, sans perte de généralité $T^0_{v_1}$, d'indice de traitement maximum $p > 1$, nous obtenons un arbre stable T^0_v avec $\texttt{pn}(T^0_v) = p$. En effet, une p-stratégie de traitement finissant en v pour T^0_v consiste en une p-stratégie de traitement pour $T^0_{v_1}$ finissant en v_1, placement d'un agent en v, retrait de l'agent positionné en v_1 et $(\leq p-1)$-stratégies de traitement pour les autres arbres stables. Donc l'Algorithme 3 retourne $(q, q') := (p, p)$. □

Théorème 9 *Étant donné un arbre $T = (V, E)$,* `Algorithme MHD` *calcule la paire $((p, p'), vect)$ associée à la décomposition hiérarchique minimale $MHD(T_v)$ de $T_v = T$ pour un certain sommet $v \in V$.*

Preuve : Nous prouvons le Théorème 9 par induction sur le nombre de sommets de l'arbre. Nous traitons tout d'abord les cas d'initialisation.

– Si v est une feuille, v ne reçoit aucun message, et donc $int_{max} = -1$. Alors l'Algorithme 3 retourne $(q, q') = (0, 0)$ et l'Algorithme 2 retourne $((0, 0), [\,])$.

– Si tous les voisins envoyant un message sont des feuilles, alors $int_{max} = 0$ et l'Algorithme 3 retourne $(q, q') = (1, 1)$ et l'Algorithme 2 retourne $((1, 1), [\,])$.

– Si v reçoit un unique message d'un sommet qui est le centre d'une étoile, alors $|I| = 1$ et $(p_i, p'_i) = (1, 1), i \in I$. L'Algorithme 3 retourne $(q, q') = (1, 2)$ et l'Algorithme 2 retourne $((1, 2), [\,])$.

– Si v reçoit un message d'un sommet qui est dans une étoile (mais pas le centre) et des messages de feuilles (d'indice de traitement 0) : $|I| = 1$ et $(p_i, p'_i) = (1, 2), i \in I$; ou si v reçoit des messages d'au moins deux sommets d'indice de traitement 1 et de sommets feuilles : $|I| \geq 2$ et $p_i = 1, i \in I$. Dans les deux cas, l'Algorithme 3 retourne $(q, q') = (2, 2)$ et l'Algorithme 2 retourne $((2, 2), [\,])$.

Dans le cas général, par l'hypothèse d'induction, v reçoit de ses fils, $v_1, \ldots, v_{d-1}$, les paires correspondantes aux décompositions hiérarchiques minimales de $T_{v_1}, \ldots, T_{v_{d-1}}$. $T_{v_1}, \ldots, T_{v_{d-1}}$, les arbres enracinés en $v_1, \ldots, v_{d-1}$, respectivement.

Étant données toutes ces décompositions hiérarchiques minimales, nous prouvons maintenant que l'Algoithme 2 calcule une décomposition hiérarchique minimale de T_v. Par le Lemme 5, l'Algorithme 3 retourne le vecteur (q, q') associé à l'arbre T_v^0 formé des arbres stables (possiblement vides) et des $(1,2)$-arbres de $MHD(T_{v_1}), \ldots, MHD(T_{v_{d-1}})$ de paires $(p_1, p'_1), \ldots, (p_{d-1}, p'_{d-1})$.

Dans une décomposition hiérarchique, nous devons respecter une hiérarchie entre les arbres. Pour garantir cela, les arbres des décompositions hiérarchiques minimales de $T_{v_1}, \ldots, T_{v_{d-1}}$ d'indices de traitement strictement plus petits que p sont ajoutés à T_v^0.

$MHD(T'_{v_1}), \ldots, MHD(T'_{v_{d-1}})$ représentent les décompositions hiérarchiques minimales obtenues à partir de $MHD(T_{v_1}), \ldots, MHD(T_{v_{d-1}})$, respectivement, gardant uniquement les arbres stables et les arbres instables d'indices de traitement strictement plus petits que q. Nous avons trois cas.

- Si $q = q'$ et s'il y a un unique $i \in [1, D-1]$ tel que $p_i = q$, alors il existe une p-stratégie de traitement pour $T_{v_i}^0 = T'_{v_i}$ finissant avec un agent sur v_i. Ainsi, nous posons un agent sur v et nous supprimons l'agent de v_i. Nous utilisons une $(\leq q-1)$-stratégie de traitement pour chaque $MHD(T'_{v_j})$ $(j \in [1, D-1], j \neq i)$. En effet, $\mathtt{pn}(T'_{v_j}) \leq q-1$ par définition.
- Si $q = q'$ et s'il y a au moins 3 arbres stables d'indice de traitement $q-1$ (il n'y a pas d'arbre d'indice de traitement q), alors nous positionnons un agent sur v, nous utilisons une $(q-1)$-stratégie de traitement pour les arbres stables d'indice de traitement $q-1$ et nous utilisons une $(\leq q-1)$-stratégie de traitement pour chaque $MHD(T'_{v_j})$ ne contenant pas les arbres stables d'indice de traitement $q-1$.
- Si $q' = q+1$, alors il y a exactement 2 arbres stables d'indice de traitement q, disons $T'_{v_1} = T^0_{v_1}$ et $T'_{v_2} = T^0_{v_2}$. Rappelons que, par définition, il n'y a pas d'arbres instables dans T'_{v_1} et T'_{v_2}. Nous utilisons une q-stratégie de traitement pour T'_{v_1} finissant avec un agent en v_1. Nous mettons un agent sur v retirant l'agent de v_1. Ensuite, nous utilisons une $(\leq q-1)$-stratégie de traitement pour chaque $MHD(T'_{v_j})$ $(j \in [3, D-1])$. Nous positionnons un agent sur v_2 supprimant l'agent de v. Enfin, nous utilisons une p-stratégie de traitement pour T'_{v_2}.

Cela est réalisé aux lignes 6 et 11 de l'Algorithme 2. L'action réalisée dans ces lignes est d'effacer les entrées de $vect_{sum}$ correspondantes aux arbres fusionnés avec T_v^0. Si T_v^0 est instable avec $\mathtt{pn}(T_b^0) = q$, alors $vect_{sum}[q]$ est incrémenté et (p, p') vaut $(-1, -1)$.

$((q, q'), vect_{sum})$ correspond à la décomposition hiérarchique minimale $MHD(T_v)$ de T_v et $MHD(T_v)$ est composé de T_v^0 et tous les autres sous-arbres instables qui ne sont pas fusionnés avec T^0 si (ligne 17) :

- $q = -1$ et $vect_{sum}$ contient seulement des 0 et des 1 ou ;
- $q \neq -1$, $vect_{sum}$ contient seulement des 0 et des 1 et la cellule q contient 0.

Sinon la décomposition hiérarchique courante n'est pas minimale car certains arbres ont le même indice de traitement. Les lignes 12 à 19 traitent de ce cas. Nous définissons k comme suit : k est la dernière cellule de $vect_{sum}$ avec un entier strictement plus grand que 1, si une telle cellule existe, sinon $k = -1$. Nous définissons k_1 comme suit : si $k \neq -1$, alors k_1 est la première cellule avec un 0 après la cellule k ; si $k = -1$, alors k_1 est la première cellule avec un 0 après la cellule q.

Nous prouvons maintenant qu'ajouter à T_v^0 tous les arbres des décompositions hiérarchiques minimales $MHD(T_{v_1}), \ldots, MHD(T_{v_{d-1}})$ dont les indices de traitement sont au plus k_1, donne un arbre stable d'indice de traitement k_1 (nous gardons la notation T_v^0 pour définir cet arbre). Pour prouver cela, nous construisons une k_1-stratégie de traitement débutant en v et montrons ensuite qu'il n'existe pas de $(k_1 - 1)$-stratégie de traitement.

Nous débutons par décrire une k_1-stratégie de traitement débutant en v. Le premier agent est donc positionné en v, la racine de T_v. La k_1-stratégie de traitement consiste à traiter séquentiellement chaque branche de T_v^0. La i^e branche est composée des arbres de $MHD(T_{v_i})$ d'indices de traitement strictement plus petits que k_1 (il n'y a pas d'arbre d'indice de traitement k_1 car $vect_{sum}[k_1] = 0$). Ces arbres forment une décomposition hiérarchique minimale de la i^e branche, et donc par le Lemme 4, il existe une $(k_1 - 1)$-stratégie de traitement. Ainsi, nous avons une k_1-stratégie de traitement pour T_v^0 car nous avons gardé l'agent positionné sur v.

Nous prouvons maitenant qu'il n'existe pas de $(k_1 - 1)$-stratégie de traitement pour T_v^0 par induction sur $k_1 - \max(k, q)$.

- $k_1 - \max(k, q - 1) = 1$:
- $\max(k, q-1) = q-1$ implique que $k_1 = q$. Mais après nous savons, par définition de q, qu'il n'existe pas de $(k_1 - 1) = (q - 1)$-stratégie de traitement pour T_v^0.
- Si $\max(k, q - 1) = k$, T_v^0 contient deux sous-arbres instables, disons T_1 et T_2, d'indices de traitement k. Ainsi, $\texttt{pn}(T_v^0) \geq k$. De plus, l'indice de traitement de $T_v^0 \setminus T_1$, qui contient T_2, a aussi un indice de traitement d'au moins k. Par le Lemme 3, comme T_v^0 contient un arbre instable T_1 d'indice de traitement k et que le reste de l'arbre a un indice de traitement plus petit que $k - 1$, nous savons que $\texttt{pn}(T_v^0) \neq k$. Comme $\texttt{pn}(T_v^0) \geq k$, nous obtenons $\texttt{pn}(T_v^0) > k$. En conséquence, il n'existe pas de $(k_1 - 1) = k$-stratégie de traitement pour T_v^0.

Le cas $\max(k, q - 1) = q - 1$ a déjà été considéré, la décomposition hiérarchique courante est déjà minimale. Si $\max(k, q - 1) = k$, l'arbre composé des deux sous-arbres instables et du chemin les joignant, a un indice de traitement de $k + 1$. En effet, par le Lemme 3, l'indice de traitement est au moins $k + 1$, et nous décrivons une $(k + 1)$-stratégie de traitement. Nous utilisons tout d'abord une $(k + 1)$-stratégie de traitement pour un des deux sous-arbres instables finissant avec un agent en sa racine. Nous utilisons ensuite une 2-stratégie de traitement pour le chemin finissant avec un agent en la racine du second sous-arbre instable. Enfin, nous utilisons une $(k+1)$-stratégie de traitement pour le second sous-arbre instable (débutant avec un agent en sa racine). Donc, $\texttt{pn}(T_v^0) \geq k_1$. Ainsi, T_v^0 est un arbre stable avec $\texttt{pn}(T_v^0) = k_1$.

- $k_1 - 1 - \max(k, q - 1) \Rightarrow k_1 - \max(k, q - 1)$: nous supposons que cela est vrai

pour $k_1-1-\max(k,q-1)$, nous prouvons que cela est vrai pour $k_1-\max(k,q-1)$. Soit T^1 l'arbre instable d'indice de traitement k_1-1. Par hypothèse d'induction, l'arbre $T_v^0 \setminus T^1$ enraciné en v a un indice de traitement de k_1-1. Ainsi, par le Lemme 3, T_v^0 a un indice de traitement de k_1.

La ligne 15 consiste à effacer de *vect* toutes les entrées correspondantes aux arbres fusionnés avec T_v^0 (ceux qui ont un indice de traitement au plus k_1-1) et la ligne 16 définit $(p,p') = (k_1,k_1)$. Nous avons une nouvelle paire $((p,p'),vect)$ avec $\texttt{pn}(T_v^0) = p$ et *vect* les arbres des décompositions hiérarchiques minimales de $T_{v_1}^0,\ldots,T_{v_d}^0$ dont les indices de traitement sont strictement plus grands que $\texttt{pn}(T_v^0) = k_1$. Par choix de k_1, il n'y a pas deux arbres d'indices de traitement identiques, alors $((p,p'),vect)$ correspond à la décomposition hiérarchique minimale $MHD(T_v)$ de T_v.

Les lignes 19 et 20 servent uniquement à satisfaire la convention qu'un vecteur de 0 est remplacé par le vecteur vide [ ]. □

2.2.4.1 Exemple d'exécution

Nous présentons un exemple d'exécution d'`Algorithme MHD` pour l'arbre T_u enraciné en u de la figure 2.9(a). T_u est composé de trois arbres de décompositions hiérarchiques minimales MHD_1, MHD_2 et MHD_3 tels que les deux racines de MHD_1 et MHD_2 sont liées via le sommet v et tel que v est lié à MHD_3 via u.

Nous supposons que le sommet v a reçu MHD_1 et MHD_2 et calcule la décomposition hiérarchique minimale $MHD(T_v)$ du sous-arbre T_v enraciné en v. Ce calcul est possible à partir des deux décompositions hiérarchiques minimales MHD_1 et MHD_2 reçues par v (figure 2.9(b)). Le vecteur *vect* de $MHD(T_v)$ est obtenu en sommant les deux vecteurs correspondant à MHD_1 et MHD_2 (voir la Table 2.2). Comme les deux arbres stables de MHD_1 et MHD_2 ont un indice de traitement de 2, nous obtenons un arbre instable dans $MHD(T_v)$ d'indice de traitement 2 (Définition 15). Nous avons ainsi $vect[2] = 1$ et $p = p' = -1$ pour la paire $((p,p'),vect)$ associée à $MHD(T_v)$ car il n'y a pas d'arbre stable dans $MHD(T_v)$. v envoie alors $MHD(T_v)$ à u.

Le sommet u calcule la décomposition hiérarchique minimale $MHD(T_u)$ de T_u à partir $MHD(T_v)$ et MHD_3 (figure 2.9(c)). En sommant les vecteurs de $MHD(T_v)$ et de MHD_3, nous obtenons le vecteur pour $HD(T_u)$ (voir la table 2.2). De plus, les paires de $MHD(T_v)$ et de MHD_3 sont $(-1,-1)$ et donnent donc un arbre stable d'indice de traitement 0 dans $HD(T_u)$. La décomposition hiérarchique minimale $MHD(T_u)$ est donnée dans la dernière ligne de la table 2.2 : un unique arbre stable avec $\texttt{pn}(T_u) = 9$.

Nous proposons dans la section 2.2.5, une version dynamique et incrémentale d'`Algorithme MHD`.

2.2.5 Algorithme dynamique et incrémental

Nous proposons dans cette section une version dynamique d'`Algorithme MHD` permettant de calculer l'indice de traitement d'un arbre obtenu par l'ajout

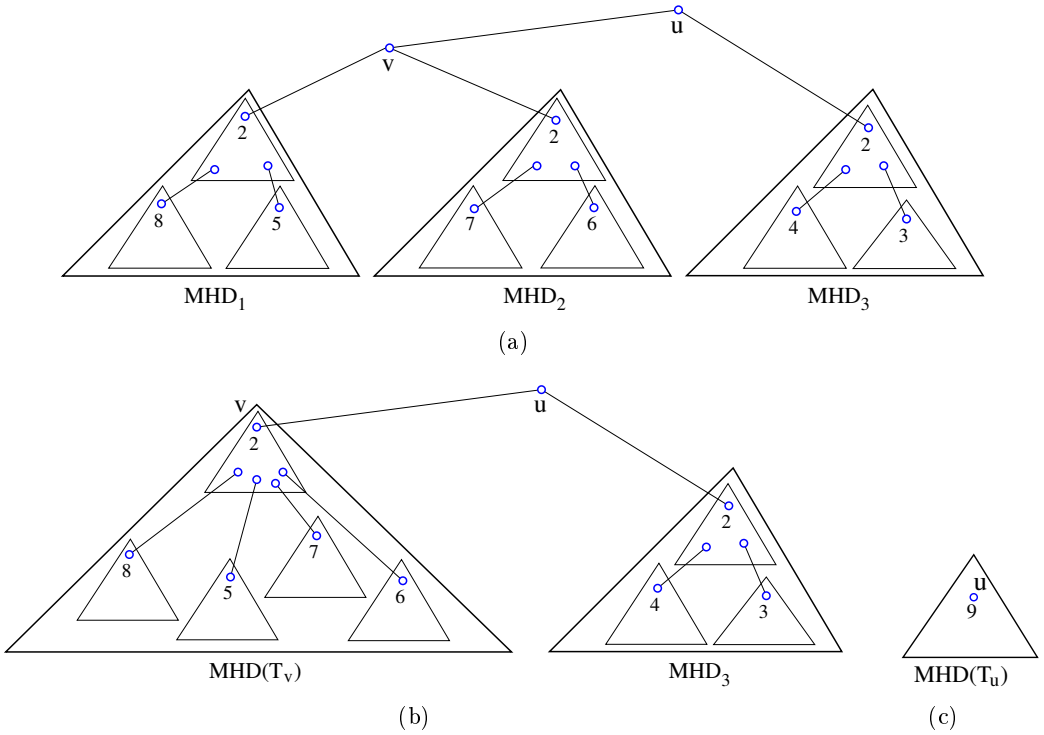


FIG. 2.9 – Exemple d'exécution d'`Algorithme MHD` pour un arbre T_u enraciné en u. (a) T_u est composé de trois décompositions hiérarchiques minimales MHD_1, MHD_2 et MHD_3 connectées via deux sommets u et v. MHD_1 et MHD_2 contiennent un arbre stable et MHD_3 contient seulement des arbres instables. Les indices de traitement des arbres sont indiqués par des entiers sur les racines ; (b) la décomposition hiérarchique minimale $MHD(T_v)$ de T_v enraciné en v est obtenue à partir de MHD_1 et de MHD_2. $MHD(T_v)$ ne contient pas d'arbre stable ; (c) la décomposition hiérarchique minimale $MHD(T_u)$ de T_u enraciné en u est obtenue à partir de $MHD(T_v)$ et de MHD_3. $MHD(T_u)$ est un unique arbre stable.

	(p,p')	$vect[1]$	$vect[2]$	$vect[3]$	$vect[4]$	$vect[5]$	$vect[6]$	$vect[7]$	$vect[8]$
MHD_1	$(2,2)$	0	0	0	0	1	0	0	1
MHD_2	$(2,2)$	0	0	0	0	0	1	1	
$MHD(T_v)$	$(-1,-1)$	0	1	0	0	1	1	1	1
MHD_3	$(-1,-1)$	0	1	1	1				
$HD(T_u)$	$(0,0)$	0	2	1	1	1	1	1	1
$MHD(T_u)$	$(9,9)$								

TAB. 2.2 – Paires $((p,p'),vect)$ associées à MHD_1, MHD_2, $MHD(T_v)$, MHD_3, $HD(T_u)$ et $MHD(T_u)$, les décompositions hiérarchiques minimales (sauf pour $HD(T_u)$) correspondantes aux arbres de la figure 2.9.

d'une arête entre deux arbres. Nous pouvons également supprimer une arête et mettre à jour le paramètre. Un des principaux avantages de l'`Algorithme MHD` est sa flexibilité. En effet, la décomposition hiérarchique nous permet de changer facilement la racine de l'arbre. Après avoir décrit les trois fonctions de la version dynamique d'`Algorithme MHD`, nous proposons un algorithme incrémental `Algorithme incrémental MHD` pour calculer l'indice de traitement d'un arbre pour lequel nous ajoutons séquentiellement les arêtes et dans n'importe quel ordre. Nous notons D le *diamètre* d'un arbre T. Le nombre d'*étapes* de ces fonctions correspond au nombre de sommets qui effectuent des calculs.

Lemme 6 (changement de racine) *Étant donnés un arbre $T=(V,E)$ de diamètre D enraciné en $r_1 \in V$ et sa décomposition hiérarchique minimale $MHD(T_{r_1})$, nous pouvons choisir une nouvelle racine $r_2 \in V$ et calculer la décomposition hiérarchique minimale $MHD(T_{r_2})$ en $O(D)$ étapes pour une complexité en temps de $O(\log n)$ pour chacune d'entre elles.*

Preuve : Nous décrivons un algorithme pour changer la racine de r_1 à r_2.

Tout d'abord, r_2 envoie un message à r_1 via le chemin unique entre r_2 et r_1, $(r_2 = u_0, u_1, u_2, \ldots, u_k = r_1)$, pour l'informer du changement. Ensuite, r_1 calcule la décomposition hiérarchique minimale $MHD(T_{r_1})$ de T_{r_1}, considérant que u_{k-1} est son parent et applique l'Algorithme 2 en utilisant $vect_{r_1}^{sum} - vect_{u_{k-1}}$ et toutes les paires précédemment reçues sauf $(p_{u_{k-1}}, p'_{u_{k-1}})$. Il envoie ensuite un message à u_{k-1}.

Après cela, u_{k-1} calcule la décomposition hiérarchique minimale $MHD(T_{u_{k-1}})$ de l'arbre $T_{u_{k-1}}$ enraciné en u_{k-1}, en supposant que u_{k-2} est son parent. Alors, u_{k-1} envoie un message à u_{k-2}. Nous répétons cela jusqu'à ce que r_2 reçoive un message de u_1. Finalement, r_2 calcule l'indice de traitement de T et devient la nouvelle racine. Nous obtenons une nouvelle décomposition hiérarchique minimale de T : $MHD(T_{r_2})$.

Dans cet algorithme, u_i soustrait le vecteur $vect_{u_{i-1}}$ de $vect_{u_i}^{sum}$, et ajoute ensuite $vect_{u_{i+1}}$, calcule (q,q') correspondant à la fusion de tous les arbres stables et des $(1,2)$-arbres des différentes décompositions hiérarchiques minimales (incluant l'arbre

stable ou le $(1,2)$-arbre de $MHD(T_{u_{i+1}})$ et soustrayant l'arbre stable ou le $(1,2)$-arbre de $MHD(T_{u_{i-1}})$) et finalement applique l'Algorithme 2. Clairement, chacun de ces calculs requiert $O(\log n)$ opérations. □

Lemme 7 (ajout d'une arête) *Soient deux arbres $T_{r_1} = (V_1, E_1)$ et $T_{r_2} = (V_2, E_2)$ enracinés en r_1 et r_2, respectivement, et leurs décompositions hiérarchiques minimales $MHD(T_{r_1})$ et $MHD(T_{r_2})$, respectivement. Nous pouvons ajouter l'arête $(w_1, w_2), w_1 \in V_1$ et $w_2 \in V_2$, et calculer l'indice de traitement de $T = (V_1 \cup V_2, E_1 \cup E_2 \cup (w_1, w_2))$, en au plus $O(D)$ étapes.*

Preuve : Nous débutons par changer les racines de T_{r_1} et T_{r_2} en w_1 et w_2, respectivement, en utilisant le Lemme 6. Ensuite w_1 ou w_2 devient la racine et calcule l'indice de traitement de T. □

Lemme 8 (suppression d'une arête) *Soient un arbre $T = (V, E)$ enraciné en r et une arête $(w_1, w_2) \in E$. Après la suppression de (w_1, w_2), si r connaît une décomposition hiérarchique minimale de T_r, nous pouvons calculer l'indice de traitement des deux arbres disjoints en au plus $O(D)$ étapes.*

Preuve : Sans perte de généralité, nous supposons que w_2 est le parent de w_1. Soient T_{w_1} le sous-arbre enraciné en w_1 et $T \setminus T_{w_1}$ l'arbre enraciné en r. Remarquons que $T \setminus T_{w_1}$ contient w_2. Étant donné que l'indice de traitement de T est calculé par l'`Algorithme MHD`, la décomposition hiérarchique minimale $MHD(T_{w_1})$ de T_{w_1} est calculée par w_1. À partir de cela, $\texttt{pn}(T_{w_1})$ peut être calculé avec le Lemme 4. Maintenant, pour calculer $\texttt{pn}(T \setminus T_{w_1})$, nous appliquons l'algorithme de changement de racine (Lemme 6) et le sommet w_2 devient la nouvelle racine de $T \setminus T_{w_1}$. □

À partir du Lemme 7, nous obtenons l'`Algorithme incrémental MHD` qui, partant d'une forêt de n sommets déconnectés avec décomposition hiérarchique minimale $((0,0),[\,])$, ajoute les arêtes de l'arbre une par une et dans n'importe quel ordre, et met à jour les indices de traitement des différentes composantes connexes. Au final, nous obtenons l'indice de traitement de T. Le nombre total de messages envoyés est $O(nD)$ et le nombre d'opérations est $O(nD\texttt{pn}(T))$. Ces valeurs sont très liées à l'ordre d'insertion des arêtes et une question intéressante est de calculer le nombre moyen de messages et d'opérations. Voir [CHM11] pour une étude du nombre d'opérations dans de bons et de mauvais cas.

2.2.6 Complexité

Nous décrivons dans cette section le nombre d'opérations effectuées par l'`Algorithme MHD` et par l'`Algorithme incrémental MHD` : accès mémoire, lecture, écriture, addition, soustraction, comparaison.

Lemme 9 *Étant donné un arbre $T = (V, E)$ avec n sommets, l'*`Algorithme MHD` *calcule* $\texttt{pn}(T)$ *en n étapes pour un total de $O(n \log n)$ opérations.*

Preuve : Chaque sommet $v \in V$ de degré d_v doit calculer int_{max} et I, qui requiert $O(d_v)$ opérations, et la somme $vect_{sum}$ de toutes les tables reçues, qui requiert $O(d_v \cdot L(vect_{sum}))$ opérations. Enfin, nous appliquons l'Algorithme 2 dans lequel toutes les opérations sont linéaires en $L(vect_{sum})$. Comme $\sum_{v \in V} d_v = 2(n-1)$ et $L(vect_{sum}) \leq \texttt{pn}(T) \leq \log_3 n$ (Théorème 8), nous avons $\sum_{v \in V} (d_v + d_v \cdot \log_3 n + \log_3 n) = O(n \log n)$. □

Le Lemme 10 décrit la taille des messages envoyés selon si la taille de l'arbre est connue ou pas.

Lemme 10 *Étant donné un arbre $T = (V, E)$ avec n sommets, l'*`Algorithme MHD` *(ou l'*`Algorithme incrémental MHD`*) envoie n messages de taille $\log_3 n + 5$ bits chacun lorsque la taille de l'arbre est connue et de taille $2\log_3 n + 6$ lorsque la taille de l'arbre n'est pas connue.*

Preuve : Chaque sommet $v \in V$ envoie $((p, p'), vect)$ correspondant à la décomposition hiérarchique minimale $MHD(T_v)$ de T_v à son parent. Nous savons par le Théorème 8 que $L(vect) \leq \log_3 n$, par la Propriété 2 que $vect$ contient seulement des 0 et des 1 et par la Remarque 3 que $vect[1] = 0$ car il n'existe pas d'arbre instable d'indice de traitement 1. Nous transmettons trois bits abc et un vecteur $vect'$ pour indiquer la valeur de $((p, p'), vect)$. Nous avons quatre codes différents :

(1) si $abc = 000$, alors $(p, p') := (-1, -1)$ et $vect := vect'$;

(2) si $abc = 001$, alors $(p, p') := (0, 0)$ et $vect := vect'$;

(3) si $abc = 010$, alors $(p, p') := (1, 1)$ et $vect := vect'$;

(4) si $abc = 011$, alors $(p, p') := (1, 2)$ et $vect := vect'$;

(5) si $abc = 100$, alors $(p, p') := (i, i)$ avec $i > 1$ la première cellule avec 1 dans $vect'$. Donc $\forall j \neq i, vect[j] := vect'[j]$ et $vect[i] := 0$;

Nous avons 3 bits pour abc et $\log_3 n - 1$ bits pour $vect'$.

De plus, nous préfixons tous les messages par 3 bits xyz afin d'indiquer quel algorithme est utilisé et si la taille de l'arbre est connue ou pas :

- $x = 1$ si la taille de l'arbre est connue et $x = 0$ sinon ;
- $y = 1$ pour `Algorithme MHD` et $y = 0$ pour `Algorithme incrémental MHD` ;
- $z = 1$ pour l'addition des vecteurs dans `Algorithme incrémental MHD` et $z = 0$ pour la soustraction des vecteurs dans `Algorithme incrémental MHD`.

En fixant z à 0 lorsque l'`Algorithme MHD` est utilisé ($y = 1$), nous gardons le préfixe $xyz = 111$ pour l'initialisation.

- Si la taille de l'arbre est connue ($x = 1$), alors chaque message est composé de xyz, abc et $vect'$ et est de taille $\log_3 n + 5$.
- Si la taille de l'arbre n'est pas connue ($x = 0$), il reste à coder le message de manière à détecter sa fin. Nous pouvons utiliser des règles simples pour coder $vect'$: nous remplaçons le bit 1 par les deux bits 11 et nous ne changons pas le bit 0. Nous ajoutons les deux bits 10 pour indiquer la fin du message. Ainsi, le message est composé de xyz, abc, $vect'$ et 10 et est de taille au plus $3 + 3 + 2L(vect') + 2$

bits. Comme $L(vect') \leq \texttt{pn}(T) - 1 \leq \log_3 n - 1$ (Théorème 8), la taille de chaque message est au plus $2\log_3 n+6$ bits. Ainsi, chaque sommet peut décoder le message reçu sans connaître n.

□

2.2.7 Extensions et perspectives

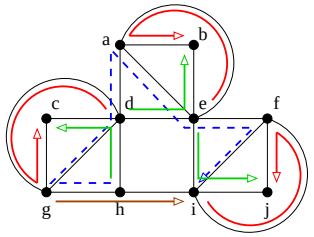
Comme mentionné précédemment, nous avons adapté `Algorithme MHD` et `Algorithme incrémental MHD` pour le calcul de l'indice d'échappement sommet, de la pathwidth, de la sommet séparation et de l'indice d'échappement arête. Pour cela, il est suffisant de changer seulement l'initialisation. Tous les détails et les preuves se trouvent dans [CHM11] (nous présentons dans cet article `Algorithme MHD` pour l'indice d'échappement sommet avant de l'étendre aux autres paramètres). Notons qu'il n'est en revanche pas possible d'adapter nos algorithmes à la version pondérée des différents problèmes, car ces derniers sont NP-difficiles [MT09], à moins que P=NP.

Nous étudions actuellement la possibilité d'adapter nos algorithmes et notre structure de décomposition hiérachique pour le calcul des différents paramètres sur des graphes ayant une structure relativement proche de l'arbre. Citons pour être précis trois classes de graphes candidates.

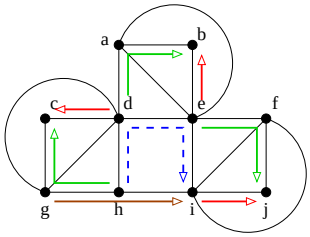
– La classe de graphes ayant un unique cycle (un arbre plus une arête formant un cycle) est une candidate naturelle. Ces graphes ont été très étudiés notamment en raison de leurs applications dans le domaine de la chimie [LZ08]. Deux algorithmes centralisés existent pour calculer la sommet séparation d'un tel graphe : en temps $O(n \log n)$ dans [EM04] et récemment en temps $O(n)$ dans [CKH06]. Il est donc intéressant de proposer un algorithme générique (distribué et centralisé) pour le calcul de tous les paramètres cités précédemment d'un graphe avec un unique cycle.

– De manière plus générale, un résultat intéressant serait d'adapter notre structure de décomposition et notre algorithme pour calculer les valeurs des différents paramètres précédemment mentionnés dans les arbres d'anneaux.

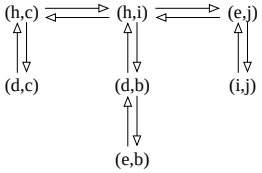
– Dans [BK96], les auteurs proposent un algorithme polynomial pour calculer la pathwidth (et donc l'indice d'échappement sommet) des graphes planaires extérieurs. Mais la complexité en $O(n^{11})$ rend ce dernier inutilisable en pratique. Un des problèmes que nous regardons est d'essayer d'adapter notre décomposition hiérarchique afin de concevoir un algorithme polynomial avec un exposant sensiblement plus petit. Cela est motivé par le fait que le graphe dual d'un graphe planaire extérieur biconnexe est un arbre. Voir [CHS07, BF02] pour plus de détails sur le lien entre la pathwidth d'un graphe planaire extérieur biconnexe et la pathwidth de son dual.



(a) Routage R.



(b) Routage R'.



(c) Digraphe de dépendances D.

FIG. 2.10 – Instance I du problème de reconfiguration : un réseau de 10 noeuds et des arcs symétriques chacun composé d'une longueur d'onde, 8 requêtes $(h,i),(h,c),(d,c),(d,b),(e,b),(e,j),(i,j),(g,i)$ à rerouter. La figure 2.10(a) représente le routage initial R, la figure 2.10(b) représente le nouveau routage R' et la figure 2.10(c) représente le digraphe de dépendances D associé à I.

2.3 Compromis entre le nombre d'interruptions simultanées et le nombre total d'interruptions

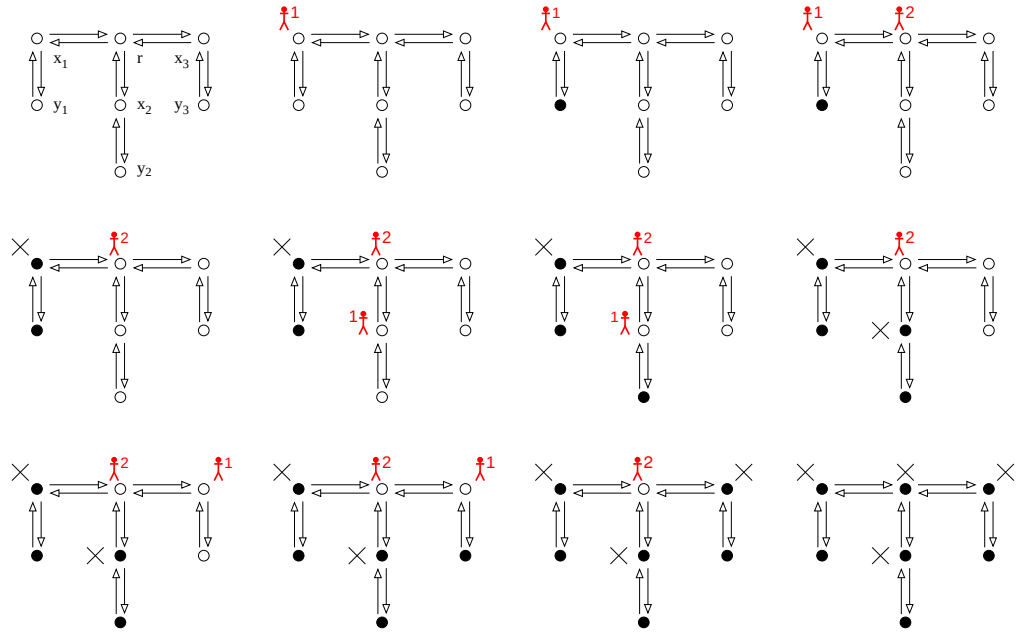
Nous nous sommes intéressés jusqu'à présent au calcul de stratégies de traitement en minimisant le nombre d'agents utilisés ou le nombre total de sommets couverts. Dans cette section, nous abordons le problème de la détermination de stratégies de traitement avec des objectifs portant simultanément sur le nombre d'agents utilisés et sur le nombre total de sommets couverts.

Définition 19 ((p,q)-stratégie de traitement pour D) *Étant donné un digraphe (de dépendances), une (p,q)-stratégie de traitement pour D est une stratégie de traitement qui utilise p agents et couvre un total de q sommets.*

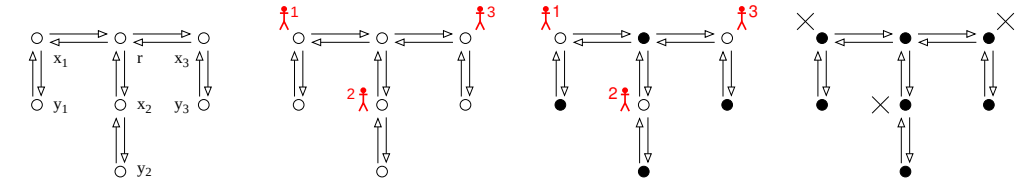
Propriété 3 *Soient I une instance du problème de reconfiguration et D le digraphe de dépendances associé à I. Il existe une reconfiguration des requêtes garantissant qu'au plus p requêtes sont interrompues simultanément avec un total de q interruptions si, et seulement si, il existe une (p,q)-stratégie de traitement pour D.*

Entre autres questions intéressantes, quelle est la plus petite valeur de q pour laquelle il existe une $(pn(D),q)$-stratégie de traitement ? Nous cherchons à déterminer si de bons compromis sont possibles en général ? Le cas échéant pour des instances particulières ?

Nous débutons par la description d'un exemple motivant cette étude. Considérons l'instance I du problème de reconfiguration représentée dans la figure 2.10. La figure 2.10(a) représente le routage courant R des huit requêtes $(h,i),(h,c),(d,c),(d,b),(e,b),(e,j),(i,j),(g,i)$. La figure 2.10(b) représente le nouveau routage R' de ces huit requêtes. Enfin, la figure 2.10(c) représente le digraphe



(a) Une $(2, 4)$-stratégie de traitement pour D.



(b) Une $(3, 3)$-stratégie de traitement pour D.

FIG. 2.11 – Deux stratégies de traitement pour le digraphe de dépendances (orienté symétrique) D de l'instance du problème de reconfiguration décrite dans la figure 2.10. La stratégie de traitement de la figure 2.11(a) est optimale en nombre d'agents et la stratégie de traitement de la figure 2.11(b) est optimale en nombre de sommets couverts.

de dépendances orienté symétrique D associé à I. La figure 2.11 représente deux stratégies de traitement pour D.

– La stratégie décrite dans la figure 2.11(a) débute par placer un agent sur le sommet x_1 (règle R_1). Cela permet le traitement de y_1 (règle R_3). Un second agent est alors placé sur r (règle R_1) permettant le traitement de x_1 et retirant l'agent placé (règle R_2). Le reste de la stratégie pour les deux autres branches est analogue. Au final, 2 agents sont utilisés pour un total de 4 sommets couverts. Cette $(2, 4)$-stratégie de traitement pour D est minimum pour p. En effet,

il n'existe pas de stratégie de traitement utilisant au plus 1 agent. Les digraphes d'indices de traitement 0 ou 1 peuvent être facilement identifiés car ils correspondent aux digraphes sans circuit et aux digraphes tels que chaque composante fortement connexe a un minimum feedback vertex set au plus 1, respectivement. Voir [CS07] et la section 4.1.3.

– Une autre stratégie pour D est décrite dans la figure 2.11(b). Cette dernière utilise 3 agents pour un total de 3 sommets couverts. Cette $(3,3)$-stratégie de traitement pour D est minimum pour q. En effet pour $i \in \{1,2,3\}$, il est possible d'observer que soit x_i soit y_i est dans tout feedback vertex set (FVS) de D à cause du circuit (x_i, y_i, x_i). De plus, la suppression de x_1, x_2 et x_3 de D est suffisante pour rendre le digraphe sans circuit. Ainsi, ces trois sommets forment un MFVS de D et donc $mfvs(D) = 3$.

En revanche, il n'existe pas de $(pn(D) = 2, mfvs(D) = 3)$-stratégie de traitement pour D, c'est-à-dire une stratégie minimisant à la fois p et q.

Nous introduisons des paramètres de compromis afin d'étudier la valeur d'un paramètre lorsque le second est contraint. En particulier, quel est le nombre minimum de sommets devant être couverts par une stratégie de traitement pour D utilisant $pn(D)$ agents ? De façon analogue, quel est le nombre minimum d'agents à utiliser par une stratégie de traitement pour D couvrant $mfvs(D)$ sommets ?

Définition 20 ($pn_q(D)$) *Étant donnés un digraphe D et un entier $q \geq mfvs(D)$, $pn_q(D)$ représente le plus petit p tel qu'une (p,q)-stratégie de traitement pour D existe. Nous écrivons $pn_{mfvs+r}(D)$ au lieu de $pn_{mfvs(D)+r}(D)$, $r \geq 0$.*

Définition 21 ($mfvs_p(D)$) *Étant donnés un digraphe D et un entier $p \geq pn(D)$, $mfvs_p(D)$ représente le plus petit q tel qu'une (p,q)-stratégie de traitement pour D existe. Nous écrivons $mfvs_{pn+r}(D)$ au lieu de $mfvs_{pn(D)+r}(D)$, $r \geq 0$.*

Intuitivement, $pn_{mfvs}(D)$ est le nombre minimum d'agents nécessaire pour une stratégie de traitement qui minimise le nombre de sommets couverts. De manière opposée, $mfvs_{pn}(D)$ est le nombre minimum de sommets qui doivent être couverts par une stratégie de traitement qui minimise le nombre d'agents. Remarquons que $pn_{mfvs}(D)$ est borné supérieurement par le MFVS maximum parmi les composantes fortement connexes de D. De plus, il est évident que $mfvs_{mfvs}(D) = mfvs(D)$ pour tout digraphe D.

Nous généralisons ce concept via la notion de *valeurs minimales* d'un digraphe.

Définition 22 (valeur minimale) *Étant donné un digraphe D, (p,q) est une valeur minimale de D si $p = pn_q(D)$ et $q = mfvs_p(D)$.*

Intuitivement, si (p,q) est une valeur minimale de D, alors diminuer strictement la valeur d'un des deux paramètres (si cela est possible) fera nécessairement augmenter l'autre. Remarquons que $(pn(D), mfvs_{pn}(D))$ et $(pn_{mfvs}(D), mfvs(D))$ sont des valeurs minimales par définition (et peuvent être les mêmes). Par exemple $(2,4)$ et $(3,3)$ sont des valeurs minimales pour le digraphe de la figure 2.11. La figure 2.12

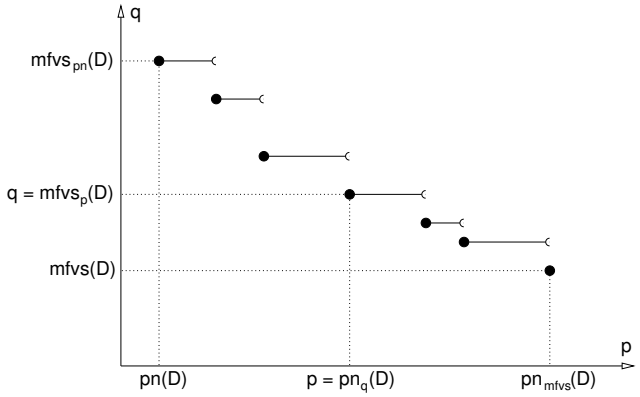


FIG. 2.12 – $mfvs_p(D)$ en fonction de p pour un digraphe D. Les cercles pleins ont pour coordonnées les valeurs minimales pour D.

décrit les variations du nombre minimum q de sommets couverts par une p-stratégie de traitement pour un digraphe D ($p \geq pn(D)$), i.e., $mfvs_p(D)$ en fonction de p. Cela représente une fonction non croissante bornée supérieurement par $mfvs_{pn}(D)$ et inférieurement par $mfvs(D)$. Les cercles de la figure 2.12 ont pour coordonnées les valeurs minimales de D. Clairement, pour un digraphe D donné, le nombre de valeurs minimales est au plus linéaire en le nombre de sommets. Nous décrivons maintenant une famille de digraphes avec n sommets pour lesquels le nombre de valeurs minimales est $\Omega(\sqrt{n})$. Soit H_n une étoile orientée symétrique avec $n \geq 3$ branches de longueur 2 (H_3 est le digraphe de la figure 2.11) et soit G_k le digraphe formé par l'union disjointe de $H_3, \ldots, H_k$, $k \geq 3$. Alors, pour tout $0 \leq i \leq k-2$, $(pn(G_k)+i, mfvs(G_k)+k-2-i) = (2+i, (k(k+1)/2)-5+k-i)$ est une valeur minimale. Nous pouvons montrer cela en généralisant la preuve pour H_3 décrite en début de section.

Nous avons analysé le comportement des deux paramètres de compromis dans les digraphes généraux et les digraphes orientés symétriques. Nous nous sommes intéressés à ce que l'on perd sur une métrique quand l'autre est fixée. En particulier, nous étudions les rapports $\frac{pn_{mfvs}(D)}{pn(D)}$ et $\frac{mfvs_{pn}(D)}{mfvs(D)}$. Cette étude comporte des Théorèmes sur la complexité pour estimer cette perte (section 2.3.1) et sur l'existence de digraphes pour lesquels elle est arbitrairement large (section 2.3.2). Enfin dans la section 2.3.3, nous étudions le cas des digraphes orientés symétriques.

Plus précisément, nous prouvons que pour tous $\alpha, \beta \geq 0$, les problèmes de déterminer les parammètres $\alpha.pn_{mfvs}(D) + \beta.pn(D)$ et $\alpha.mfvs_{pn}(D) + \beta.mfvs(D)$ sont NP-complets (Théorème 10, section 2.3.1). En particulier, le problème de déterminer $pn_{mfvs}(D)$ n'est pas approximable à une constante additive près en temps polynomial (à moins que P=NP) et celui de déterminer $mfvs_{pn}(D)$ est APX-difficile (Théorème 10, section 2.3.1). Nous prouvons aussi que pour tout $q \geq 0$ (respectivement pour tout $p \geq 0$), le rapport $\frac{pn_{mfvs+q}(D)}{pn(D)}$ (respectivement $\frac{mfvs_{pn+p}(D)}{mfvs(D)}$) n'est pas borné même pour la classe des digraphes d'indices de traitement bor-

nés (Théorème 11 et Théorème 12, section 2.3.2). Cependant, nous prouvons que $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq pn(D)$ pour tout digraphe orienté symétrique D (Lemme 11, section 2.3.3).

2.3.1 Les paramètres de compromis sont difficiles à approximer

Nous étudions dans cette section la complexité des problèmes de calculer les différents paramètres de compromis définis précédemment. Nous définissons tout d'abord quelques digraphes qui seront utilisés dans les différentes preuves.

Soit H_n une étoile orientée symétrique composée de $n \geq 3$ branches chacune contenant 2 sommets (la racine r est à distance 2 de toute feuille). Il y a donc $2n+1$ sommets. H_3 est représenté dans la figure 2.11. Il est facile de vérifier que $pn(H_n) = 2$. En effet, 1 agent n'est clairement pas suffisant et il existe de plus une $(2, n+1)$-stratégie de traitement pour H_n. Premièrement, un agent est placé sur la racine r. Ensuite, un agent est placé sur un sommet x adjacent à r, l'autre voisin de x est traité et x est traité (l'agent positionné sur x est donc retiré). Nous appliquons cette séquence pour tous les autres voisins de r. Nous traitons finalement r en retirant l'agent placé au tout début de la stratégie de traitement. La figure 2.11(a) représente une $(2,4)$-stratégie de traitement pour H_3. De plus, l'unique MFVS de H_n est l'ensemble X des n sommets adjacents à r. Il est ensuite possible de vérifier que l'unique stratégie de traitement occupant seulement les sommets de X consiste à placer n agents sur tous les sommets de X, de traiter tous les autres sommets et enfin de traiter les sommets de X en retirant les agents. Notons qu'aucun agent ne peut être retiré tant que tous les agents ne sont pas placés. Cela représente une (n,n)-stratégie de traitement, et $pn_{mfvs}(H_n) = n$. La figure 2.11(b) représente une telle stratégie de traitement pour H_3. En résumé,

Propriété 4 *Les deux valeurs minimales de H_n sont $(pn(H_n), mfvs_{pn}(H_n)) = (2, n+1)$ et $(pn_{mfvs}(H_n), mfvs(H_n)) = (n,n)$).*

Soit K_n le digraphe orienté symétrique complet composé de n sommets. Il est facile de vérifier que l'unique valeur minimale de K_n est $(pn(K_n), mfvs(K_n)) = (n-1, n-1)$.

Soit $D = (V, A)$ un digraphe orienté symétrique avec $V = \{u_1, \ldots, u_n\}$. Soit $\hat{D} = (V', A')$ le digraphe orienté symétrique avec $V' = V \cup \{v_1, \ldots, v_n\}$ et $\hat{D}$ est construit à partir de D en ajoutant deux arcs symétriques entre u_i et v_i pour $i = 1, \ldots, n$. Il est possible de vérifier qu'il existe une stratégie de traitement optimale pour $\hat{D}$ telle que l'ensemble des sommets occupés est V. En effet, pour tout i, au moins un des deux sommets u_i ou v_i doit être couvert par un agent (tout FVS de D contient au moins v_i ou u_i). De plus si une étape d'une stratégie de traitement pour $\hat{D}$ consiste à placer un agent sur v_i, alors la stratégie de traitement peut être transformée afin de placer un agent sur u_i à la place. En particulier, $mfvs_{pn}(\hat{D}) = n$.

Théorème 10 *Soient $\alpha, \beta \geq 0$ fixés, avec $\max\{\alpha, \beta\} > 0$. Le problème qui prend un digraphe D en entrée et qui consiste à déterminer :*

– *$\alpha.pn_{mfvs}(D) + \beta.pn(D)$ n'est pas approximable à une constante additive près en temps polynomial (à moins que P=NP),*
– *$\alpha.mfvs_{pn}(D) + \beta.mfvs(D)$ est APX-difficile.*

Preuve : Les deux cas pour $\alpha = 0$ et $\beta > 0$ proviennent clairement de la littérature. Nous supposons donc que $\alpha > 0$.

– Nous débutons avec $\alpha.pn_{mfvs}(D) + \beta.pn(D)$.

Considérons tout d'abord le cas $\beta = 0$. Nous montrons alors que le problème de déterminer pn_{mfvs} n'est pas approximable à une constante additive près en temps polynomial, à moins que P=NP. En effet, soit $\mathcal{D}$ la classe des digraphes $\hat{D}$ obtenus à partir des digraphes orientés symétriques D. Pour tout digraphe orienté symétrique D, le problème de calculer $pw(D)$ (avec $pw(D)$ la pathwidth du graphe sous-jacent au graphe orienté symétrique D) n'est pas approximable à une constante additive près en temps polynomial (à moins que P=NP), et $pn(\hat{D}) = pn_{mfvs}(\hat{D}) = pw(D)+1$ (voir le Théorème 4). Le problème de déterminer pn_{mfvs} n'est donc pas approximable à une constante additive près en temps polynomial, à moins que P=NP.

Supposons maintenant que $\beta > 0$. Pour prouver que déterminer $\alpha.pn_{mfvs}(D) + \beta.pn(D)$ n'est pas approximable à une constante additive près en temps polynomial (à moins que P=NP), soit D_1 l'union disjointe de H_n et de n'importe quel digraphe D composé de n sommets. Par le Théorème 5, notons que $pn_{mfvs}(D_1) = pn_{mfvs}(H_n)$ car $pn_{mfvs}(D) \leq n-1$ et $pn_{mfvs}(H_n) = n$. Comme $pn(D_1) = \max\{pn(D), pn(H_n)\}$ et $pn(H_n) = 2$, nous obtenons que $\alpha.pn_{mfvs}(D_1) + \beta.pn(D_1) = \alpha.n + \beta\max\{pn(D), 2\}$. Le problème de déterminer l'indice de traitement n'étant pas approximable à une constante additive près en temps polynomial (à moins que P=NP), nous en déduisons le résultat.

– Nous considérons maintenant $\alpha.mfvs_{pn}(D) + \beta.mfvs(D)$.

Lorsque $\beta = 0$, nous prouvons que le problème de déterminer $mfvs_{pn}$ est APX-difficile. Soit D_2 l'union disjointe de K_n et de n'importe quel digraphe D composé de n sommets. Par le Théorème 5, $pn(D_2) = \max\{pn(K_n), pn(D)\}$. Il est facile d'observer que $pn(D_2) = pn(K_n) = n-1$ car $pn(D) \leq n-1$. Ainsi, lorsque D doit être traité, $n-1$ agents sont disponibles. Donc pour minimiser le nombre de sommets couverts par les agents, ces derniers doivent être placés sur un MFVS de D. Donc $mfvs_{pn}(D_2) = n-1+mfvs(D)$ et le résultat est démontré car calculer $mfvs(D)$ est un problème APX-difficile.

Supposons maintenant que $\beta > 0$. Pour prouver que $\alpha.mfvs_{pn}(D) + \beta.mfvs(D)$ est APX-difficile, soit D_3 l'union disjointe de K_n, H_n, et D. Nous avons $pn(D_3) = \max\{pn(K_n), pn(H_n), pn(D)\}$. Observons que $pn(D_3) = pn(K_n) = n-1$ car $pn(H_n) = 2$ et $pn(D) \leq n-1$. De plus, toute stratégie de traitement pour D_3 utilisant $n-1$ agents doit couvrir $n-1$ sommets de K_n, $n+1$ sommets de H_n ($mfvs(H_n) = n$ mais un agent supplémentaire est nécessaire pour couvrir seulement n sommets), et $mfvs(D)$ sommets de D (car $n-1$ agents sont disponibles et $mfvs(D) \leq n-1$). Ainsi, $mfvs_{pn}(D_3) = (n-1)+(n+1)+mfvs(D)$. De plus $mfvs(D_3) = (n-1)+n+mfvs(D)$ car $mfvs(K_n) = n-1$ et $mfvs(H_n) = n$.

Donc $\alpha.mfvs_{pn}(D_3) + \beta.mfvs(D_3) = (\alpha+\beta)(mfvs(D)+2n) - \beta$. Le résultat est prouvé car le problème de calculer $mfvs(D)$ est APX-difficile.

□

Corollaire 3 *Étant donnés un digraphe D et deux entiers $p \geq 0$ et $q \geq 0$, et pour tout $\alpha, \beta \geq 0$ ($\{\alpha, \beta\} \neq \{0, 0\}$) le problème de déterminer :*

- *$\alpha.pn_{mfvs+q}(D)+\beta.pn(D)$ n'est pas approximable à une constante additive près en temps polynomial (à moins que P=NP),*
- *$\alpha.mfvs_{pn+p}(D) + \beta.mfvs(D)$ est APX-difficile.*

2.3.2 Les rapports de compromis ne sont pas bornés dans les digraphes

Dans cette section, nous étudions le comportement des paramètres de compromis et leurs rapports. Nous montrons notamment qu'en général, de bons compromis sont impossibles.

Théorème 11 *$\forall C > 0$ et $\forall q \geq 0$, il existe un digraphe D tel que $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$.*

Preuve : Soit l'étoile orientée symétrique décrite dans la section 2.3.1. Soit D le digraphe composé de $q+1$ copies disjointes de H_n. D a $q+1$ composantes fortement connexes. Nous obtenons que $mfvs(D) = (q+1)n$. Par définition, toute $(pn_{mvfs+q}(D), mfvs(D)+q)$-stratégie de traitement pour D couvre au plus $q(n+1)+n$ sommets. Donc il existe au moins une des $q+1$ composantes fortement connexes pour laquelle au plus n sommets sont couverts. Ainsi, pour traiter cette composante, n agents sont nécessaires. En effet, (n, n) est une valeur minimale pour H_n, et par définition nous ne pouvons pas diminuer la première valeur sans augmenter la seconde. En conclusion, $pn_{mfvs+q}(D) = n$ alors que $pn(D) = 2$. En choisissant $n > 2C$, nous obtenons $\frac{pn_{mfvs+q}(D)}{pn(D)} > C$. □

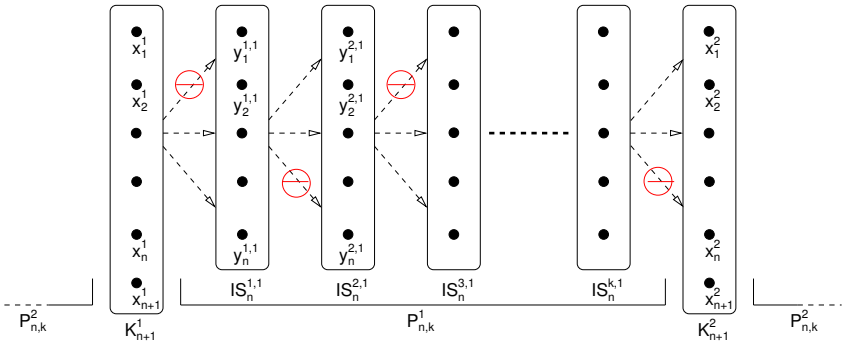
Observons que si le nombre de sommets pouvant être couverts est $mfvs(D)+q+1$ (au lieu de $mfvs(D)+q$), alors le nombre minimum d'agents nécessaires est $pn(D)$. En d'autres termes, pour le digraphe D décrit dans la preuve du Théorème 11, nous obtenons $\frac{pn_{mfvs+q+1}(D)}{pn(D)} = 1$ alors que $\frac{pn_{mfvs+q}(D)}{pn(D)} = \frac{n}{2}$.

Corollaire 4 *$\forall C > 0$, il existe un digraphe D tel que $\frac{pn_{mfvs}(D)}{pn(D)} > C$.*

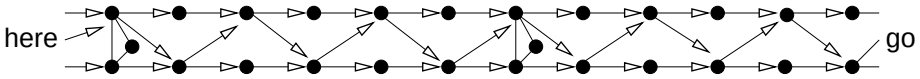
Nous présentons maintenant des résultats similaires pour le second rapport de compromis.

Théorème 12 *$\forall C > 0$ et $\forall p \geq 0$, il existe un digraphe D tel que $\frac{mfvs_{pn+p}(D)}{mfvs(D)} > C$.*

Esquisse de la preuve : Soit $n \geq 2$ et soit $k \geq 1$ un entier impair. Considérons le digraphe $D_{n,k}$ construit comme suit. Soient $IS_n^1, \cdots, IS_n^k$ k ensembles indépendants, chaque IS_n^t $(1 \leq t \leq k)$ est composé de n sommets : $y_1^t, y_2^t, \ldots, y_n^t$. Soit $P_{n,k}$ le digraphe obtenu à partir des k ensembles indépendants IS_n^t $(1 \leq t \leq k)$ en ajoutant



(a) $D_{n,k}$ du Théorème 12 et du Corollaire 5 (k impair). Le symbole rouge $\ominus$ représente la non existence d'arcs entre les sous-graphes. Les arcs de $V(D_{n,k}) \setminus V(K^1_{n+1})$ à $V(K^1_{n+1})$ ne sont pas représentés.



(b) $D_{2,5}$ du Corollaire 5. Les arcs de tous les sommets vers les trois sommets de K^1_3 ne sont pas représentés.

FIG. 2.13 – Digraphe $D_{n,k}$ décrit dans le Théorème 12 et dans le Corollaire 5.

les arcs de y^t_i à y^{t+1}_j, pour $1 \leq j \leq i \leq n$ et $t = 1, 3, \ldots, k-2$, et de y^t_i à y^{t+1}_j, pour $1 \leq i \leq j \leq n$ et $t = 2, 4, \ldots, k-1$. Soit K_{n+1} un digraphe complet orienté symétrique de $n+1$ sommets : $x_1, x_2, \ldots, x_{n+1}$.

Le digraphe $D_{n,k}$ est obtenu à partir de deux copies $P^1_{n,k}, P^2_{n,k}$ de $P_{n,k}$ et deux copies K^1_{n+1}, K^2_{n+1} de K_{n+1}, en ajoutant les arcs suivants. Dans ce qui suit, $y^{t,a}_j$ représente le j^e sommet dans le t^e ensemble indépendant de $P^a_{n,k}$, avec $j \leq n$, $t \in \{1, k\}$, $a \in \{1, 2\}$, et x^a_j représente le j^e sommet de K^a_n, avec $j \leq n+1$, $a \in \{1, 2\}$. Il y a des arcs de x^a_i à $y^{1,a}_j$, pour $1 \leq i \leq j \leq n$ et $a = 1, 2$, et de $y^{k,a}_i$ à x^b_j, pour $1 \leq i \leq j \leq n$, $a = 1, 2$ et $b = 3 - a$. Enfin il y a un arc de chaque sommet de $V(D_{n,k}) \setminus V(K^1_{n+1})$ à chaque sommet de $V(K^1_{n+1})$. Remarquons que ces derniers arcs ne sont pas nécessaires pour obtenir le résultat mais rendent la preuve plus digeste.

La figure 2.13(a) représente $D_{n,k}$ (le symbole rouge $\ominus$ représente les non-arcs entre ces sous-graphes). $D_{2,5}$ est représenté dans la figure 2.13(b). Afin de ne pas surcharger les figures, les arcs de $V(D_{n,k}) \setminus V(K^1_{n+1})$ à $V(K^1_{n+1})$ ne sont pas représentés.

Clairement, $mfvs(D_{n,k}) = 2n$ et tout MFVS est composé de $\{x^1_1, \ldots, x^1_n\}$ et de n sommets de K^2_{n+1}. Nous prouvons que $pn(D_{n,k}) \leq n+1$. Soit p, $0 \leq p \leq n-2$ (nous choisissons en fait $n \geq p+2$). Nous prouvons ensuite que toute stratégie de traitement pour $D_{n,k}$ utilisant $n+1+p$ agents, couvre au moins un sommet de chacun des k ensembles indépendants de $P^1_{n,k}$. La preuve est assez technique et le lecteur

intéressé peut la trouver dans [CCM+ar]. En conclusion, en prenant $k > 2n(C-1)$, nous obtenons $\frac{mfvs_{pn+p}(D_{n,k})}{mfvs(D_{n,k})} = \frac{mfvs_{pn+p}(D_{n,k})}{2n} \geq \frac{mfvs_{n+1+p}(D_{n,k})}{2n} \geq \frac{2n+k}{2n} > C$. □

Remarquons qu'il existe une $(pn(D)+p+1, mfvs(D))$-stratégie de traitement pour le digraphe $D_{n,k}$ décrit dans la preuve du Théorème 12 alors que le plus petit q tel qu'il existe une $(pn(D)+p, q)$-stratégie de traitement pour $D_{n,k}$, est arbitrairement grand.

Corollaire 5 *$\forall C > 0$, il existe un digraphe D tel que $\frac{mfvs_{pn}(D)}{mfvs(D)} > C$.*

Nous obtenons ce résultat en considérant le digraphe $D_{n,k}$ décrit dans la figure 2.13(a), avec $n = 2$ et $k \geq 1$ (la figure 2.13(b) représente $D_{2,5}$). Ce digraphe est tel que $pn(D_{2,k}) = 3$ et $mfvs(D_{2,k}) = 4$ alors que $\frac{mfvs_{pn}(D_{2,k})}{mfvs(D_{2,k})} = \frac{k+4}{4}$ n'est pas borné.

Le Lemme 11 de la section 2.3.3 montre que, dans la classe des graphes orientés symétriques à indice de traitement borné, $\frac{mfvs_{pn}(D)}{mfvs(D)}$ est borné.

2.3.3 Rapports de compromis dans les digraphes orientés symétriques

Dans cette section, nous étudions le comportement du rapport $\frac{mfvs_{pn}(D)}{mfvs(D)}$ dans les digraphes orientés symétriques D. Rappelons que les rapports $\frac{pn_{mfvs+q}(D)}{pn(D)}$ et $\frac{pn_{mfvs}(D)}{pn(D)}$ ont déjà été étudiés dans la section 2.3.2 pour des digraphes orientés symétriques à indice de traitement borné.

Lemme 11 *Pour tout digraphe orienté symétrique D, $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq pn(D)$.*

Preuve : Sans perte de généralité, nous prouvons le résultat pour un digraphe fortement connexe $D = (V, E)$. Soit S une $(pn(D), mfvs_{pn}(D))$-stratégie de traitement pour D. Soit $O \subseteq V$ l'ensemble des sommets couverts par un agent durant l'exécution de S. Soit F un MFVS de D. Nous effectuons une partition de V en $(Y, X, W, Z) = (O \cap F, O \setminus F, F \setminus O, V \setminus (O \cup F))$. Comme D est orienté symétrique, $V \setminus F$ est un ensemble indépendant car c'est le complémentaire d'un MFVS. Comme les sommets non couverts durant S ont tous leurs voisins couverts, $V \setminus O$ est un ensemble indépendant. Étant donné $V' \subseteq V$, $N(V')$ est l'ensemble des voisins des sommets de V'. La partition est représentée dans la figure 2.14.

Tout d'abord, notons que $|N(W) \cap X| \leq pn(D)|W|$. En effet pour qu'un sommet $v \in W$ soit traité, tous ses voisins doivent être occupés par un agent. Ainsi, le degré maximum de v est $pn(D)$.

Nous prouvons ensuite que $|X \setminus N(W)| \leq (pn(D)-1)|Y|$. Soit $R = X \setminus N(W)$. Comme $X \cup Z$ est un ensemble indépendant, pour chaque $v \in R$, $N(v) \subseteq Y$. Soit $T = N(R) \subseteq Y$. Remarquons que $N(T) \cap R = R$ car D est fortement connexe et orienté symétrique. Nous ordonnons les sommets de $T = \{v_1, \cdots, v_t\}$ dans l'ordre de traitement (lorsque les agents sont retirés) durant l'exécution de S. Pour tout i, $1 \leq i \leq t$, soit $N_i = \bigcup_{j \leq i} N(v_j) \cap R$. Nous prouvons maintenant que $|N_1| < pn(D)$

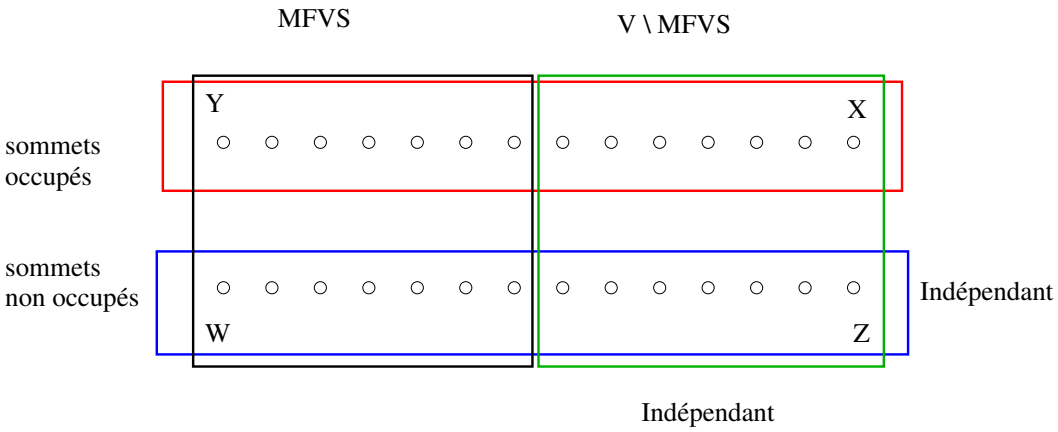


FIG. 2.14 – Preuve du Lemme 11.

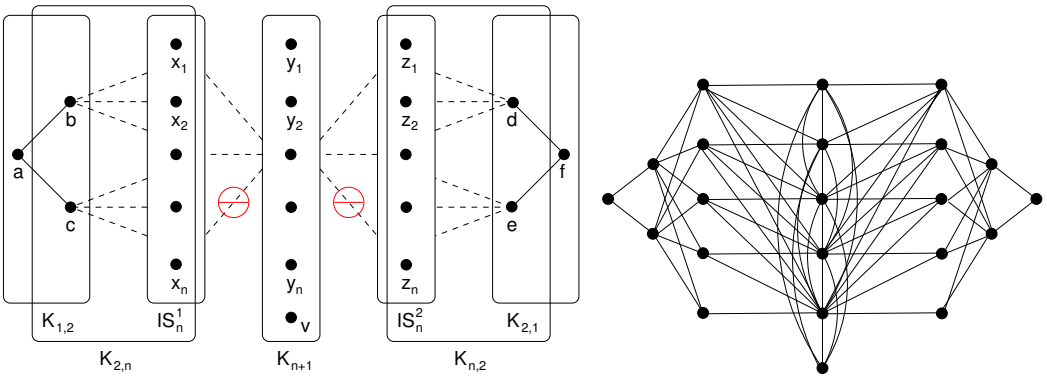
et que $|N_{i+1} \setminus N_i| < pn(D)$ pour tout $i < t$. Nous obtiendrons alors $|N_t| = |R| \leq (pn(D) - 1)|T| \leq (pn(D) - 1)|Y|$.

Considérons l'étape de S juste avant qu'un agent soit retiré de v_1. Soit $v \in N_1 \neq \emptyset$. Comme l'agent sera retiré de v_1, soit v a déjà été traité, soit il est occupé par un agent. Nous montrons qu'il existe un sommet dans $N(v) \subseteq T$ qui n'a pas encore été occupé et donc que v doit être occupé. En effet, sinon, tous les voisins de v sont occupés (car, à cette étape, aucun agent n'a été retiré des sommets de T) et la stratégie peut traiter v sans placer d'agent dessus, contredisant le fait que S occupe le plus petit nombre de sommets possible. Donc, juste avant qu'un agent soit retiré de v_1, tous les sommets de N_1 sont occupés par un agent. Nous obtenons que $|N_1| < pn(D)$.

Soit $1 < i \leq t$. Considérons l'étape de S juste avant qu'un agent soit retiré de v_i. Soit $v \in N_i \setminus N_{i-1}$ si un tel sommet existe. Comme l'agent sera retiré de v_i, soit v a déjà été traité, soit il est occupé par un agent. Nous montrons qu'il y a un sommet dans $N(v) \subseteq T \setminus N_{i-1}$ qui n'a pas encore été occupé par un agent et que v doit être occupé. En effet, sinon, tous les voisins de v sont occupés (comme, à cette étape, aucun agent n'a été retiré des sommets de $T \setminus N_{i-1}$) et la stratégie peut traiter v sans positionner un agent dessus, contredisant le fait que S occupe le plus petit nombre de sommets possible. Donc, juste avant qu'un agent soit retiré de v_i, tous les sommets de $N_{i+1} \setminus N_i$ sont occupés par un agent. Nous obtenons que $|N_{i+1} \setminus N_i| < pn(D)$.

En conclusion, $mfvs_{pn}(D) = |O| = |Y| + |X|$ et $|X| = |X \setminus N(W)| + |N(W) \cap X|$. Ainsi, $mfvs_{pn}(D) \leq pn(D)(|Y| + |W|) = pn(D)|F| = pn(D).mfvs(D)$. □

Nous montrons dans le Lemme 12 qu'il existe des digraphes orientés symétriques avec un indice de traitement aussi proche de 3 que nous le souhaitons, avant de conjecturer que cette classe de digraphes a un indice de traitement borné par 3



(a) SD_n (b) SD_5

FIG. 2.15 – Digraphe orienté symétrique SD_n du Lemme 12 (figure 2.15(a)) et instance de SD_n avec $n = 5$ (figure 2.15(b)). Le symbole rouge $\ominus$ représente la non existence d'arcs.

(Conjecture 1). La preuve complète du Lemme 12 se trouve dans [CCM+ar].

Lemme 12 $\forall \varepsilon > 0$, *il existe un digraphe orienté symétrique* D *tel que* $3 - \varepsilon \leq \frac{mfvs_{pn}(D)}{mfvs(D)} < 3$.

Esquisse de la preuve : Soit $n \geq 1$. Nous construisons le digraphe SD_n comme suit. Soient IS_n^1 et IS_n^2 deux ensembles indépendants composés de n sommets chacun : $x_1, \ldots, x_n$ et $z_1, \ldots, z_n$, respectivement. Soit K_{n+1} un digraphe complet symétrique de $n+1$ sommets $y_1, \ldots, y_n, y_{n+1} = v$. Le digraphe SD_n est construit à partir de l'union disjointe de IS_n^1, IS_n^2, K_{n+1} et de 6 sommets isolés $\{a, b, c, d, e, f\}$ en ajoutant les arcs suivants. Il y a des arcs symétriques entre les sommets x_i et y_j et les sommets z_i et y_j, pour tout $1 \leq i \leq j \leq n$. De plus, tous les arcs symétriques du digraphe biparti avec les partitions $\{b, c\}$ et IS_n^1 sont ajoutés. De manière analogue, tous les arcs symétriques du digraphe biparti avec les partitions $\{d, e\}$ et IS_n^2 sont ajoutés. Enfin, les arcs symétriques $(a, b), (a, c), (d, f), (e, f)$ sont ajoutés. Le digraphe SD_n est représenté dans la figure 2.15(a). La figure 2.15(b) décrit le digraphe SD_5.

Remarquons maintenant que $F = \{y_1, \ldots, y_n, b, c, d, e\}$ est un minimum feedback vertex set de SD_n, avec $|F| = n + 4$. En effet, tout feedback vertex set de SD_n contient au moins n sommets de K_{n+1}. Il est ensuite facile de vérifier que quatre autres sommets sont nécessaires pour former un feedback vertex set. Nous avons donc $mfvs(SD_n) = n + 4$. De plus, nous avons clairement que $pn(SD_n) \geq n$. Nous montrons ensuite que toute statégie utilisant $n + 1$ agents, couvre au moins $3n + 2$ sommets. La preuve est assez technique et le lecteur intéressé peut la trouver dans [CCM+ar]. Nous décrivons une $(n + 1, 3n + 2)$-stratégie de traitement pour SD_n. Comme $mfvs_{pn}(SD_n) \geq mfvs_{n+1}(SD_n) = 3n + 2$, nous obtenons que $\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} \geq \frac{3n+2}{n+4}$. En choisissant $n > \frac{10}{\varepsilon} - 4$, nous obtenons que

$\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} \geq 3 - \varepsilon$. De plus, comme SD_n est composé de $3n + 7$ sommets, nous avons que $mfvs_{pn}(SD_n) \leq 3n + 6$, et donc que $\frac{mfvs_{pn}(SD_n)}{mfvs(SD_n)} < 3$. □

Conjecture 1 *Pour tout digraphe orienté symétrique D, $\frac{mfvs_{pn}(D)}{mfvs(D)} \leq 3$.*

2.3.4 Perspectives

Nous avons étudié dans cette section les problèmes consistant à minimiser un paramètre lorsque l'autre est fixé. Nous avons prouvé que les problèmes de calculer ces différents paramètres de compromis sont APX-difficiles pour certains et ne sont pas approximables à une constante additive près (à moins que P=NP) pour d'autres. Nous avons également mis en exergue le fait que pour certains digraphes, minimiser un paramètre peut augmenter la valeur de l'autre arbitrairement. Cela reste vrai même si nous nous accordons un nombre quelconque d'agents supplémentaires ou un nombre quelconque de sommets supplémentaires autorisés à être couverts, respectivement.

Il est intéressant de continuer ces recherches pour tenter de prouver ou d'infirmer la Conjecture 1 concernant les digraphes orientés symétriques. De plus, il serait intéressant de concevoir des algorithmes bi-critrères exacts et heuristiques propres au calcul de (p, q)-stratégies de traitement, c'est-à-dire avec des objectifs portant sur les deux paramètres.

Une problèmatique importante pour la reconfiguration est de prendre en compte le temps. Par exemple, le nombre d'étapes durant lesquelles un sommet est couvert par un agent peut être important car cela peut correspondre, dans un certain sens, à la durée d'interruption de la requête correspondante. Un problème est alors de minimiser le nombre d'étapes durant lesquelles le sommet de couverture maximum est couvert. Le sommet de couverture maximum est le sommet le plus longtemps couvert au cours de la stratégie. L'étude de possibles compromis entre tous ces paramètres semble donc très intéressante d'un point de vue théorique et avec des conséquences concrètes dans la reconfiguration du routage. Voir [Cou10] pour plus de détails sur la modélisation avec prise en compte du temps. Coudert y introduit, entre autres, le problème de minimisation du nombre d'étapes des stratégies de traitement.

2.4 Extensions et autres modélisations

Dans cette dernière section, nous présentons une extension de notre modèle ainsi que d'autres modélisations relatives à la reconfiguration du routage des connexions. Dans la section 2.4.1, nous introduisons la notion de requêtes prioritaires. Ces dernières ne peuvent pas être interrompues au cours de la reconfiguration. Nous montrons comment se ramener au problème initial sur un digraphe de dépendances transformé. Dans la section 2.4.2, nous supposons que toute longueur d'onde peut être partagée par plusieurs connexions. Nous montrons alors, entre autres, que déterminer s'il existe une reconfiguration sans interruption est un problème NP-complet.

Rappelons que ce problème est très facile dans le modèle initial. Dans la section 2.4.3, nous nous intéressons à la prise en compte de contraintes physiques liées par exemple à certains réglages à effectuer lorsqu'une connexion change de route. Nous ne prenons plus en compte les interruptions liées aux dépendances. En effet, nous supposons que les ressources sont suffisantes pour effectuer le reroutage sans interruption.

2.4.1 Modèle avec requêtes prioritaires

Comme nous l'avons vu jusqu'à présent, la reconfiguration du routage peut engendrer des interruptions temporaires de certaines requêtes. Ces interruptions peuvent être inacceptables pour certains clients. En effet, ces derniers peuvent avoir contracté des clauses spécifiques interdisant toute interruption du trafic. C'est pourquoi nous introduisons dans notre modèle une contrainte supplémentaire qui interdit toute interruption de trafic pour chacune des requêtes appartenant à un sous-ensemble donné Q de requêtes dites *prioritaires*. En d'autres termes, étant donnés une instance I et un ensemble Q, le problème revient à déterminer une reconfiguration avec la contrainte supplémentaire que chacune des requêtes de Q est reroutée en utilisant un make-before-break. De manière équivalente, cela revient à déterminer une stratégie de traitement pour le digraphe de dépendances D associé à I avec la contrainte qu'un agent ne peut pas être placé sur un sommet $u \in Q$. Nous notons Q l'ensemble des sommets de D correspondant à l'ensemble Q des requêtes prioritaires. Nous notons (I, Q) une instance du problème de reconfiguration contraint par Q et nous notons (D, Q) le digraphe de dépendances avec les sommets prioritaires Q. Définissons formellement la notion de (p, q)-stratégie de traitement pour (D, Q).

Définition 23 ((p,q)-stratégie de traitement pour (D,Q)) *Étant donnés un digraphe de dépendances $D = (V, A)$ et un ensemble de sommets prioritaires $Q \subseteq V$, une (p, q)-stratégie de traitement pour (D, Q) est une (p, q)-stratégie de traitement pour D telle que les sommets de l'ensemble Q ne sont pas couverts par un agent.*

La propriété 5 montre à nouveau l'équivalence entre les notions de reconfiguration et de stratégie de traitement.

Propriété 5 *Soient I une instance du problème de reconfiguration, $D = (V, A)$ le digraphe de dépendances associé et Q un sous-ensemble de requêtes (sommets) prioritaires. Il existe une reconfiguration des requêtes garantissant qu'aucune requête de Q ne soit interrompue et qu'au plus p requêtes sont interrompues simultanément avec un total de q interruptions si, et seulement si, il existe une (p, q)-stratégie de traitement pour (D, Q).*

Ainsi, un sommet $u \in Q$ doit être traité en utilisant la règle R_3, c'est-à-dire sans placer d'agent sur u. Si le sous-digraphe induit par les sommets de Q n'est pas sans circuit, alors il n'existe pas de stratégie de traitement valide, et donc pas de reconfiguration du routage valide (Lemme 13). Décider cela peut se faire linéairement en la somme du nombre de sommets et du nombre d'arcs (Proposition 2).

Lemme 13 *Soient $D = (V, A)$ un digraphe de dépendances et $Q \subseteq V$ un sous-ensemble de sommets prioritaires. Il n'existe pas de stratégie de traitement pour (D, Q) si et seulement si D contient un circuit de sommets prioritaires.*

Preuve : $\Leftarrow$ Si D contient un circuit de sommets prioritaires, il n'est pas possible d'utiliser la règle R_3 pour traiter un sommet du circuit sans placer préalablement un agent sur au moins un autre sommet de ce circuit.

$\Rightarrow$ Si D ne contient pas de circuit de sommets prioritaires, alors une stratégie de traitement pour (D, Q) peut consister à premièrement placer un agent sur chacun des sommets de $V - Q$, traiter ensuite les sommets de Q (sans agent supplémentaire) et enfin traiter les sommets de $V - Q$ en retirant les agents placés dessus. □

Proposition 2 *Étant donnés un digraphe de dépendances $D = (V, A)$ et un ensemble de sommets prioritaires $Q \subseteq V$, nous pouvons décider en temps $O(|V| + |A|)$ si une stratégie de traitement existe pour (D, Q).*

Preuve : Par le Lemme 13, il suffit de tester si le digraphe D_Q, obtenu en supprimant les sommets $V - Q$ de D, contient une composante fortement connexe. □

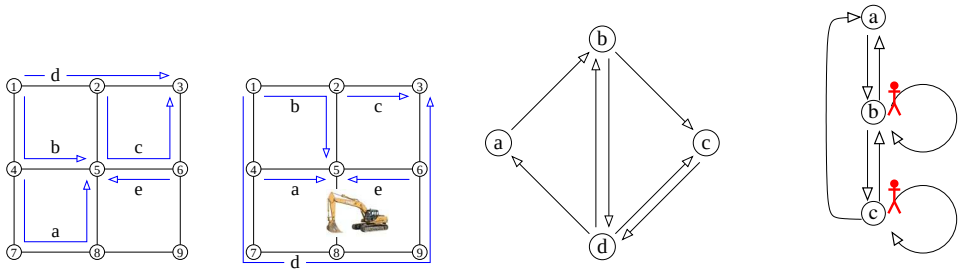
Dans le cas où le digraphe induit par Q est sans circuit, nous montrons dans la section 2.4.1.1 qu'il existe une (p, q)-stratégie de traitement pour (D, Q) si, et seulement si, il existe une (p, q)-stratégie de traitement pour D^*, construit à partir de D en temps polynomial. En d'autres termes, nous montrons que le problème de reconfiguration avec requêtes prioritaires se ramène à notre problème initial de reconfiguration.

2.4.1.1 Transformation

Nous allons construire un digraphe D^* à partir de D et Q, tel que s'il existe une (p, q)-stratégie de traitement pour D^* alors il existe une (p, q)-stratégie de traitement pour (D, Q), et réciproquement.

Définition 24 (construction de D*(v)) *Soient $D = (V, A)$ un digraphe et $v \in V$. Le digraphe $D^*(v)$ est obtenu à partir de D en supprimant v et en ajoutant un arc de tous les sommets de $N^-(v)$ vers tous les sommets de $N^+(v)$. Remarquons que cette transformation peut générer des boucles si $N^-(v) \cap N^+(v) \neq \emptyset$.*

Considérons l'instance de la figure 2.16. Chaque lien du réseau représente deux arcs (un dans chaque sens) chacun composé d'une longueur d'onde. Le routage courant R des cinq requêtes a, b, c, d et e est représenté dans la figure 2.16(a). Le routage final R' est représenté dans la figure 2.16(b) et le digraphe de dépendances D associé est représenté dans la figure 2.16(c). Nous supposons que la requête d est prioritaire ($Q = \{d\}$). En utilisant la transformation décrite dans la Définition 24, nous obtenons le digraphe $D^* = D^*(d)$ représenté dans la figure 2.16(d). Comme $N^-(d) = \{b, c\}$, $N^+(d) = \{a, b, c\}$, et donc $N^-(v) \cap N^+(v) = \{b, c\}$, les sommets b et c ont des boucles dans $D^*(d)$. Comme nous le prouvons dans la Proposition 3,



(a) Routage R. (b) Routage R'. (c) Digraphe de dépendances D. (d) Construction de $D^* = D^*(d)$ à partir de D et $Q = \{d\}$.

FIG. 2.16 – Le réseau est une grille 3×3 avec des liens symétriques de capacité 1 (c'est-à-dire composé d'une longueur d'onde). Les figures 2.16(a) et 2.16(b) représentent le routage courant R et le routage final R' pour l'ensemble de connexions $\{a, b, c, d, e\}$, respectivement. L'ensemble des requêtes prioritaires est $Q = \{d\}$. La figure 2.16(c) est le digraphe de dépendances associé et la figure 2.16(d) représente $D^* = D^*(v)$ construit à partir de D (Définition 24).

l'existence d'une $(2,2)$-stratégie de traitement pour D^* (optimal pour les deux paramètres) assure une $(2,2)$-stratégie de traitement pour $(D, \{d\})$. En effet, il est nécessaire pour traiter D^* de placer deux agents sur les deux sommets avec boucle (sommets b et c), et cela simultanément. Le sommet a peut ainsi être traité. Enfin, nous traitons les sommets b et c séquentiellement.

Proposition 3 *Soient $D = (V, A)$ un digraphe et $Q := \{v\}$ pour un certain sommet $v \in V$ sans boucle. Une (p,q)-stratégie de traitement pour (D, Q) existe si et seulement si une (p,q)-stratégie de traitement pour $D^*(v)$ existe.*

Preuve : Soit P une (p,q)-stratégie de traitement pour (D, Q). Nous appliquons P pour $D^*(v)$ en ignorant l'étape de traitement du sommet v. Il ne peut y avoir un problème uniquement lorsque nous traitons un sommet $u \in N_D^-(v)$. Mais lorsque une telle étape survient, le sommet v a en fait été traité précédemment car P est une stratégie de traitement valide. En conséquence, tous les sommets de $N_D^+(v)$ sont déjà traités ou couverts par un agent, et le sommet u peut être traité. Donc il existe une (p,q)-stratégie de traitement pour $D^*(v)$.

Considérons maintenant P^* une (p,q)-stratégie de traitement pour $D^*(v)$. Nous obtenons P à partir de P^* en ajoutant l'étape consistant à traiter v juste avant le premier traitement d'un sommet de $N^-(v)$. Par définition de $D^*(v)$, tous les sommets de $N^+(v)$ doivent être déjà traités ou couverts par un agent. En conséquence, nous obtenons que P est une (p,q)-stratégie de traitement valide pour (D, Q). □

Plus généralement, en appliquant récursivement la technique de la Proposition 3, étant donnés un digraphe de dépendances $D = (V, A)$ et un ensemble $Q \subseteq V$ de som-

mets prioritaires, nous pouvons construire un digraphe D^* sans sommet prioritaire et déduire une (p,q)-stratégie de traitement pour (D,Q) à partir d'une (p,q)-stratégie de traitement pour D^* (Corollaire 6).

Corollaire 6 *Soient $D = (V,A)$ un digraphe et $Q := \{v_1, v_2, \ldots, v_f\} \subseteq V$. Nous définissons $D_1 := D^*(v_1)$ et pour chaque $i \in \{2,3,\ldots,f\}$, nous définissons $D_i := D^*_{i-1}(v_i)$ si v_i n'a pas de boucle dans V_i. Une (p,q)-stratégie de traitement pour (D,Q) existe si et seulement si $D^* := D_f$ est défini et une (p,q)-stratégie de traitement pour D_f existe.*

Remarque 7 *Observons que si v_{i+1} a une boucle dans V_i, alors D contient un circuit de sommets prioritaires et donc D ne peut pas être traité sans placer au moins un agent sur un sommet prioritaire. Il n'existe donc pas de reconfiguration n'interrompant pas de requête prioritaire.*

Le Corollaire 6 met en exergue, entre autres, que déterminer l'indice de traitement de D^* permet de déduire le nombre minimum d'agents nécessaires pour traiter D avec les sommets prioritaires Q. En d'autres termes, nous avons $\mathtt{pn}(D,Q) = \mathtt{pn}(D^*)$ avec $\mathtt{pn}(D,Q)$ le plus petit p tel qu'il existe une (p,q)-stratégie de traitement pour (D,Q).

2.4.1.2 Borne supérieure sur le nombre d'agents supplémentaires

Nous prouvons maintenant une borne supérieure sur le nombre nécessaire d'agents supplémentaires pour traiter un digraphe $D = (V,A)$ avec l'ensemble $Q \subseteq V$ de sommets prioritaires. En d'autres termes, nous comparons $\mathtt{pn}(D)$ et $\mathtt{pn}(D,Q) = \mathtt{pn}(D^*)$.

Proposition 4 *Pour tout digraphe $D = (V,A)$ et pour tout sous-ensemble $Q \subseteq V$ de sommets prioritaires, $\mathtt{pn}(D,Q) \leq \mathtt{pn}(D) + |N^+(Q)|$. De plus, cette borne est asymptotiquement atteinte.*

Preuve : Soit P une $\mathtt{pn}(D)$-stratégie de traitement pour D. Nous appliquons la stratégie de traitement P pour (D,Q) avec les adaptations suivantes. À chaque étape de P qui consiste à placer un agent sur un sommet $v \in Q$, nous remplaçons cette opération par la séquence d'opérations qui consiste à placer un agent sur chaque sommet de $N^+(v)$. Si un sommet de $N^+(v)$ est un sommet prioritaire, nous plaçons aussi un agent sur chaque sommet de son voisinage sortant et ainsi de suite récursivement. Ensuite, tous les sommets prioritaires considérés à cette étape sont traités. Si à une étape de la stratégie de traitement, P place un agent sur un sommet déjà occupé (cela peut arriver si ce sommet est un voisin sortant d'un sommet prioritaire), alors nous passons simplement cette étape. Enfin, si P traite un sommet qui n'est pas occupé mais qui est occupé par un agent dans la stratégie de traitement modifiée, alors cet agent est supprimé. Cette nouvelle stratégie de traitement est bien une stratégie de traitement pour (D,Q) car lorsqu'un sommet

v de D est occupé par un agent dans P, alors soit c'est également le cas dans la nouvelle stratégie, soit v est un sommet prioritaire et a été traité. De plus, la stratégie de traitement modifiée utilise au plus $\mathtt{pn}(D) + |N^+(Q)|$ agents.

Pour prouver que cette borne est atteinte, soit $k \geq 1$. Considérons le digraphe D composé de k cliques orientées symétriques de taille k : $C_1, C_2, \cdots, C_k$ et de $k+1$ sommets $v_0, \cdots, v_k$. Nous avons un arc de v_0 à v_i pour tout $i \leq k$. Ensuite, pour tout $i \leq k$, nous avons un arc de v_i à chaque sommet de C_i et de chaque sommet de C_i à chaque sommet de C_{i+1}. Enfin, nous avons un arc de chaque sommet de $\bigcup_{i \leq k} C_i$ à v_0. Nous prouvons premièrement que $\mathtt{pn}(D) = k+1$: la stratégie de traitement consiste à placer un agent sur v_0, traiter $v_1, \cdots, v_k$ et enfin placer un agent succesivement sur chaque sommet de C_i, pour i allant de 1 à k. Pour conclure, soit $Q = \{v_0, \cdots, v_k\}$. Nous prouvons que $\mathtt{pn}(D, Q) = \sum_{i \leq k} |C_i|$. En appliquant la transformation décrite précédemment pour les sommets $v_1, \cdots, v_k$, nous obtenons un digraphe dans lequel le sommet prioritaire v_0 est un voisin entrant de tous les sommets de $\bigcup_{i \leq k} C_i$. Par la Proposition 3, nous obtenons le résultat. □

Nous avons présenté une modélisation du problème de reconfiguration avec la contrainte que certaines requêtes prioritaires ne peuvent pas être interrompues (clause dans le contrat par exemple). Nous représentons cela en introduisant dans les (p, q)-stratégies de traitement la contrainte que les sommets prioritaires ne peuvent pas recevoir d'agents. Il est très simple de vérifier si une stratégie de traitement existe pour un digraphe et un ensemble de sommets prioritaires donnés. Dans ce cas, nous proposons une transformation du digraphe permettant de nous ramener au problème de déterminer une (p, q)-stratégie de traitement sans contrainte de sommets prioritaires. En définitive, il n'est pas utile de traiter ce problème indépendamment et tous les résultats présentés pour les digraphes de dépendances valent pour les digraphes de dépendances avec sommets prioritaires.

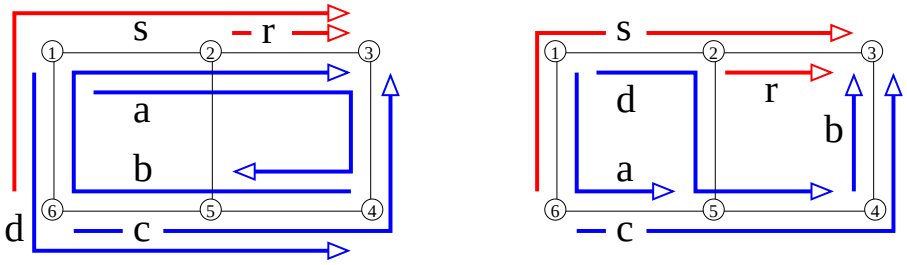
2.4.2 Modèle avec partage de la longueur d'onde

Dans le modèle que nous avons utilisé jusqu'à présent, les relations de dépendances entre les requêtes étaient binaires. En effet, étant donnés une instance I du problème de reconfiguration, le digraphe de dépendances $D = (V, A)$ associé à I et deux sommets (requêtes) $u \in V$ et $v \in V$ devant être reroutées, nous avions que :

- soit $(u, v) \in A$ et donc u pouvait être reroutée si et seulement si v était interrompue ou déjà reroutée ;
- soit $(u, v) \notin A$ et donc le reroutage de u pouvait se faire indépendamment du reroutage de v.

Dans [CMN09b], nous avons relâché la contrainte qu'une longueur d'onde pouvait être utilisée par au plus une requête.

La figure 2.17 représente un réseau en grille avec des liens symétriques. Il y a une longueur d'onde sur chacun des liens. Chacune de ces longueurs d'ondes peut être partagée par n'importe quelle paire de connexions. La figure 2.17(a) représente un routage valide R des requêtes a, b, c et d mais ne permet pas d'accepter les deux nouvelles requêtes s et r sans changer les routes de R. Nous pouvons véri-



(a) Routage R des requêtes a, b, c, d. Les nouvelles requêtes r et s ne peuvent pas être routées.

(b) Routage R' des requêtes a, b, c, d, r, s.

FIG. 2.17 – Le réseau est une grille avec des liens symétriques. Il y a une longueur d'onde sur chacun des liens. Chacune de ces longueurs d'ondes peut être partagée par deux connexions. Le routage R (figure 2.17(a)) ne permet pas l'établissement des connexions des nouvelles requêtes r et s. Il est alors possible d'utiliser le routage R' (figure 2.17(b)).

fier que chaque arc du réseau supporte au plus 2 connexions. Le routage R' de la figure 2.17(b) représente un routage valide des 6 requêtes. La requête d peut être reroutée dès que l'une des requêtes a ou b est interrompue (ou déjà reroutée). En effet, la nouvelle route de la requête d dans R' utilise l'arc $(1, 2)$ et les deux requêtes a et b utilisent cet arc dans le routage initial R. Ainsi, il suffit qu'une de ces deux requêtes libèrent cette ressource pour permettre le reroutage de d.

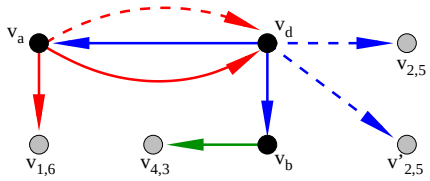
Une reconfiguration sans interruption consiste à rerouter séquentiellement la requête b, puis la requête d et la requête a. Il est ensuite possible d'établir les connexions des requêtes r et s. Notons que la requête c ne change pas de route.

Nous modélisons ces *dépendances partielles* par un *multi-digraphe de dépendances* et définissons la notion de *stratégie de traitement généralisée* (section 2.4.2.1). Nous montrons ensuite que le problème de décider s'il existe une reconfiguration sans interruption est NP-complet (section 2.4.2.2).

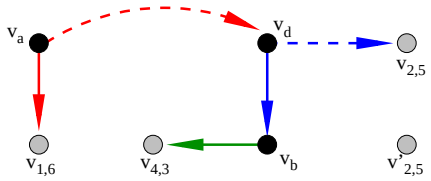
2.4.2.1 Multi-digraphe de dépendances et stratégie de traitement généralisée

Le réseau est modélisé par un digraphe $G = (V, E)$ avec une fonction capacité $c : E \to \mathbb{N}$ sur les arcs de G et un ensemble Υ de requêtes de connexion. Un *routage* R est *valide* si pour chaque arc $e \in E$, e appartient à au plus $c(e)$ chemins de R.

Le *problème de reconfiguration généralisée* consiste à rerouter les requêtes de connexion les unes après les autres d'un routage initial R à un routage final R'. Une requête r peut être reroutée de sa route initiale dans R vers sa nouvelle route dans R' seulement si la capacité est disponible tout au long du nouveau chemin. Clairement, il est possible que certaines requêtes doivent être reroutées avant r pour rendre disponible la capacité nécessaire. Ces contraintes sont modélisées par un



(a) Multi-digraphe de dépendances D correspondant au reroutage des requêtes de R à R'.



(b) Possible reconfiguration sans interruption.

FIG. 2.18 – Multi-digraphe de dépendances D et possible reconfiguration. Les arcs de la même couleur représentent les dépendances concernant le lien e et donc les arcs étiquetés e. Les sommets virtuels sont représentés en gris.

multi-digraphe de dépendances.

Le multi-digraphe de dépendances $D = (W, A)$ avec des étiquettes sur les arcs est construit comme suit. Il y a un sommet v_r dans W pour chaque requête $r \in \Upsilon$ devant changer de route. De plus, pour modéliser la capacité disponible $q(e)$ d'un lien $e \in E$ dans le routage R ($q(e)$ égal $c(e)$ moins le nombre de requêtes utilisant e dans R), il y a un sommet dans W pour chaque unité de capacité disponible sur le lien e. Soit $\mathcal{V}(e) = \{v_e^1, \ldots, v_e^{q(e)}\}$ l'ensemble de ces $q(e)$ sommets. Nous notons $\mathcal{V} = \cup_{e \in E} \mathcal{V}(e)$ l'ensemble de tels sommets que nous appelons *sommets virtuels*. Il y a un arc étiqueté e du sommet u vers le sommet v s'il y a un lien e de G appartenant à $R'(u)$ et v est un sommet virtuel dans $\mathcal{V}(e)$, ou si $e \in R'(u) \cap R(v)$ avec v corespondant à une requête. En d'autres termes, tous les arcs sortants d'un sommet u étiqueté e représentent les ressources possibles qui peuvent être utilisées par la requête u pour traverser le lien e dans sa route finale.

Par exemple dans la figure 2.17, les routes des requêtes a, b et d sont différentes dans R' et dans R, cela permettant de router les requêtes r et s dans R'. Le multi-digraphe de dépendances D, représenté dans la figure 2.18(a), admet $\{v_a, v_b, v_d, v_{1,6}, v_{4,3}, v_{2,5}, v'_{2,5}\}$ comme ensemble de sommets, avec v_a correspondant à la requête a et $v_{i,j}$ correspondant à une unité de capacité disponible sur le lien (i, j) (2 unités de capacité disponibles sur le lien $(2, 5)$). Il y a des arcs de même étiquette dans D de v_a à v_d et de v_a à $v_{1,6}$ car, sur le lien $(1, 6)$, la requête a peut utiliser dans R' soit l'unité de capacité disponible, $v_{1,6}$, soit l'unité de capacité qui sera disponible si d est reroutée avant a.

Nous pouvons remarquer sur la figure 2.18(a) que la requête b peut être reroutée à tout moment car son seul voisin sortant (dans D) est une unité de capacité disponible. De plus, la requête d a besoin d'une unité de capacité disponible sur le lien $(2, 5)$, soit $v_{2,5}$ soit $v'_{2,5}$, et elle a besoin d'une unité de capacité sur le lien $(1, 2)$. Comme les deux seules unités de capacité sont utilisées par les requêtes a et b, alors soit la requête a soit la requête b doit être reroutée avant d.

Nous énonçons à présent quelques remarques.

– Le degré sortant d'un sommet virtuel dans D est 0 (voir la figure 2.18(a)).

– Pour tout lien $e \in E(G)$, soit D_e le sous-graphe de D induit par tous les arcs

étiquetés e. Ainsi, $D_e = (X \cup Y, F)$ est le digraphe complet biparti des requêtes qui utiliseront e dans R' (les sommets dans X) vers les requêtes qui utilisent e dans R et les éventuels sommets virtuels correspondant (les sommets dans Y). Notons que $|X| \leq |Y|$. Nous appelons un tel graphe un *beau digraphe biparti*.

La principale difficulté vient du fait que le routage final R' indique seulement les liens utilisés par les requêtes mais pas les unités de capacités. Nous avons potentiellement de nombreuses possibilités lors de la reconfiguration.

Plus précisément, lorsque nous reroutons une requête r vers sa route finale dans R', pour tout $e \in R'(r)$, nous devons décider quelle unité de capacité de e sera utilisée par r. Cela correspond à choisir l'arc sortant de r étiqueté e qui sera utilisé dans le sous-graphe D_e.

En d'autres termes, pour tout $e \in E(G)$, une reconfiguration donne un couplage maximal de D_e (rappelons que l'ensemble des sommets de D_e est noté $X \cup Y$). En effet, pour toute étiquette des arcs, exactement un arc sortant avec cette étiquette est choisie pour toute requête dans X et deux arcs de même étiquette ne peuvent pas être incidents au même sommet de Y car cela voudrait dire que deux requêtes utiliseraient la même ressource pour traverser le lien e. Réciproquement, pour tout arête $e \in E(G)$ (i.e., pour toute étiquette des arcs de D), soit M_e un couplage maximal de D_e. Alors, le sous-graphe de D induit par $\bigcup_{e \in E(G)} M_e$ correspond à une reconfiguration compatible avec le routage R' et les capacités des liens du réseau. Par exemple, la figure 2.18(b) représente un choix d'un tel couplage pour le multi-digraphe de dépendances décrit dans la figure 2.18(a).

Soit D un multi-digraphe avec des étiquettes sur les arcs, telles que les arêtes avec la même étiquette e induisent un beau digraphe biparti D_e (nous disons que D est *bien étiqueté*). Pour toute étiquette e, soit $\mathcal{M}_e$ l'ensemble des couplages maximaux de D_e. Soit $\mathcal{D}$ l'ensemble de tous les digraphes qui peuvent être obtenus en choisissant un couplage maximal M_e de $\mathcal{M}_e$ pour chaque étiquette e et en considérant le sous-graphe induit par $\bigcup_e M_e$.

Définition 25 *L'*indice de traitement généralisé *de D, noté* $\texttt{gpn}(D)$*, est le plus petit indice de traitement de D' parmi tous les D' dans $\mathcal{D}$.*

Par définition, l'indice de traitement généralisé d'un multi-digraphe de dépendances de deux routages R et R' est égal au plus petit nombre de requêtes qui doivent être simultanément interrompues durant la reconfiguration de R à R'.

Notons que l'indice de traitement généralisé est un invariant de multi-digraphe avec des étiquettes sur les arcs. En particulier, si tous les arcs de D ont des étiquettes différentes (lorsque tous les liens de G ont capacité 1), l'indice de traitement généralisé de D est égal à son indice de traitement. En effet, dans ce cas, tout sous-graphe D_e induit par les arcs de D avec la même étiquette e ($e \in E(G)$) est une unique arête, et donc $\mathcal{D} = \{D\}$.

2.4.2.2 Décider s'il existe une reconfiguration sans interruption est NP-complet

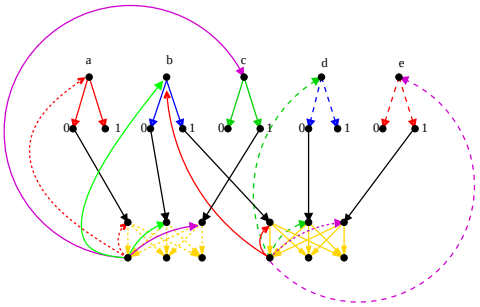
Le Théorème 13 montre la grande difficulté du problème de reconfiguration généralisée. En effet, le problème de décider s'il existe une reconfiguration des requêtes de connexion sans interruption de trafic est NP-complet alors que dans le modèle initial, ce problème de décision est très simple. En d'autres termes, le problème de décider si $\texttt{gpn}(D) = 0$ est NP-complet.

Théorème 13 *Si une longueur d'onde peut être partagée par au plus trois requêtes, le problème de décider s'il existe reconfiguration sans interruption est NP-complet.*

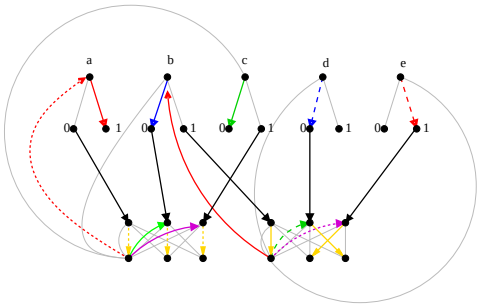
Preuve : Le problème appartient clairement à NP car $\texttt{gpn}(D) = 0$ si et seulement s'il existe un couplage maximal M_e pour toute étiquette e, tel que le sous-graphe induit par $\bigcup_e M_e$ est un DAG.

La preuve réduit 3-SAT à notre problème. Soit F une formule 3-SAT avec les variables $x^1, \cdots, x^n$ et les clauses $C^1, \cdots, C^m$. De F nous construisons un digraphe D composé de $3n + 6m$ sommets avec des étiquettes sur les arêtes tel que, pour toute étiquette, le sous-graphe induit par les arêtes avec cette étiquette est soit une étoile de degré sortant maximum trois soit un $K_{3,3}$. Nous construisons D comme suit (voir la figure 2.19(a) pour un exemple) :

1. pour chaque variable x^i de F, nous ajoutons trois sommets v^i, v^i_0, et v^i_1 ;
2. pour tout $i \leq n$, nous choisissons une étiquette pas utilisée et nous ajoutons deux arcs avec cette étiquette de v^i à v^i_0 et de v^i à v^i_1. Intuitivement, tout couplage maximal dans le sous-graphe induit par ces deux arcs représente une assignation de la variable x^i ;
3. pour chaque clause C^j de F, nous ajoutons les six sommets a^j_1, a^j_2, a^j_3, c^j, c^j_0 et c^j_1. Intuitivement, a^j_1, a^j_2 et a^j_3 seront associés aux variables apparaissant dans C^j, et c^j représentera la valeur de C^j ;
4. pour tout $j \leq m$, nous choisissons une étiquette pas utilisée et nous ajoutons tous les arcs avec cette étiquette de a^j_1, a^j_2 et a^j_3 à c^j, c^j_0 et c^j_1. Notons que c^j_0 et c^j_1 sont utilisés seulement afin d'assurer que le sous-graphe obtenu avec cette étiquette soit un beau digraphe biparti ;
5. pour tout $j \leq m$, considérons une variable x^i qui apparaît dans C^j. Nous choisissons une étiquette pas utilisée et nous ajoutons un arc soit de v^i_0 à a^j_1 si l'occurence x^i apparaît dans C^j soit de v^i_1 à a^j_1 si l'occurence $\bar{x}^i$ apparaît dans C^j. Nous disons que nous *associons* x^i à a^j_1. De manière analogue, en utilisant des étiquettes distinctes, nous associons la deuxième variable apparaissant dans C^j avec a^j_2, et la troisième variable avec a^j_3 ;
6. pour tout $j \leq m$ et pour toute variable x^i qui apparaît dans C^j et associée à a^j_ℓ, $\ell \in \{1, 2, 3\}$, nous choisissons une étiquette pas utilisée et nous ajoutons un arc de c^j à v^i et de c^j à a^j_ℓ.



(a) Réduction de la formule $(a \vee b \vee \neg c) \wedge (\neg b \vee d \vee \neg e)$.



(b) Ensemble de couplages induisant un DAG. Les arcs en gris ont été "supprimés".

FIG. 2.19 – Réduction d'une formule 3-SAT à notre problème (figure 2.19(a)) et ensemble de couplages formant un DAG et donc l'assignation correspondante (figure 2.19(b)).

Notons que pour tout arc e défini à l'étape 5, e est le seul arc avec son étiquette et donc que tout couplage maximal contient e.

Soit F une formule qui ne peut pas être satisfaite. Soit D le digraphe obtenu à partir de F avec la construction décrite précédemment. Pour toute étiquette e des arcs de D, nous choisissons un couplage maximal arbitraire M_e du sous-graphe induit par les arêtes d'étiquette e. Nous prouvons que le sous-graphe H de D induit par $\bigcup_e M_e$ contient un circuit. Nous considérons l'assignation β telle que pour toute variable x^i, nous posons $\beta(x^i) = vrai$ si H contient l'arc (v^i, v^i_1) et $\beta(x^i) = faux$ sinon (H contient l'arc (v^i, v^i_0)). Comme F ne peut pas être satisfaite, il existe un $j \leq m$, tel que $\beta(C^j) = faux$. Dans H, le sommet c^j est la tête d'un arc et sans perte de généralité, nous supposons que $(a^j_1, c^j) \in E(H)$. Soit x^i la variable apparaissant dans C^j qui est associée à a^j_1. Nous prouvons le résultat en supposant que C^j contient l'occurence de $\bar{x}^i$, l'autre cas est similaire. Par définition de D, $(v^i_1, a^j_1) \in E(D)$ et c'est le seul arc avec son étiquette. Donc, $(v^i_1, a^j_1) \in E(H)$. Comme $\beta(C^j) = faux$, alors $\beta(x^i) = vrai$ et $(v^i, v^i_1) \in E(H)$. Pour conclure, il est suffisant de remarquer que soit (c^j, v^j) soit (c^j, a^j_1) est un arc de H. Ainsi, H contient un circuit : (c^j, a^j_1, c^j) ou $(c^j, v^i, v^i_1, a^j_1, c^j)$.

Soit β une assignation satisfaisant F. Nous montrons que, pour toute étiquette, nous pouvons choisir un couplage maximal avec cette étiquette tel que le sous-graphe induit par ces arcs est sans circuit. Pour tout $i \leq n$, nous choisissons l'arc (v^i, v^i_1) si $\beta(x^i) = vrai$, et (v^i, v^i_0) sinon. Pour tout $j \leq m$, considérons la clause C^j. Il existe un $i \leq n$ tel que, soit $\beta(x^i) = vrai$ et il y a une occurence de x^i dans C^j, ou $\beta(x^i) = faux$ et il y a une occurrence de $\bar{x}^i$ dans C^j (un tel entier i existe car β est une assignation satisfaisant F). La variable x^i joue un rôle particulier pour la clause C^j car elle implique $\beta(C^j) = vrai$. Sans perte de généralité, nous supposons que x^i est associé à a^j_1 dans D. Nous choisissons (a^j_1, c^j), (a^j_2, c^j_0) et (a^j_3, c^j_1) et nous choisissons $(c^j, v^i), (c^j, a^j_2)$ et (c^j, a^j_3). Voir la figure 2.19(b) pour un exemple.

Nous prouvons maintenant que le sous-graphe H de D est sans circuit.

Nous prouvons premièrement qu'aucun circuit ne passe par v^i pour toute variable x^i. Notons que v^i a degré sortant 1 dans H et que H contient soit (v^i, v^i_1) soit (v^i, v^i_0), sans perte de généralité, $(v^i, v^i_0) \in E(H)$. Par construction de H, cela veut dire que $\beta(x^i) = faux$. Si v^i_0 est une feuille (i.e., a degré sortant 0), aucun circuit ne passe par v^i. Sinon, considérons un voisin sortant a^j_ℓ de v^i_0, pour un $j \leq m, \ell \in \{1,2,3\}$. Nous prouvons que le seul voisin sortant de a^j_ℓ a degré sortant 1 et que ce n'est pas v^i, et donc qu'aucun circuit ne passe par v^i. En effet, (v^i_0, a^j_ℓ) appartient à $E(D)$ seulement s'il existe une occurrence de x^i dans C^j (étape 5 de la construction de D). Comme $\beta(x^i) = faux$, la valeur de x^i n'implique pas que $\beta(C^j) = vrai$ et, par construction de H, $(a^j_\ell, c^j) \notin E(H)$. Donc, l'unique voisin sortant de a^j_ℓ est soit c^j_0 soit c^j_1 qui ont un degré sortant 0. De plus, pour tout $i \leq n$, le seul voisin entrant de v^i_0 (et de v^i_1) dans H, s'il existe, est v^i. Donc aucun circuit ne passe par v^i_p.

Par construction de D, pour tout $j < k \leq m$, il n'y a pas d'arcs entre un sommet de $\{a^j_1, a^j_2, a^j_3, c^j, c^j_0, c^j_1\}$ et un sommet de $\{a^k_1, a^k_2, a^k_3, c^k, c^k_0, c^k_1\}$. Ainsi, s'il existe un circuit dans H, alors il y a un $j \leq m$, tel que les sommets de ce circuit appartiennent à $\{a^j_1, a^j_2, a^j_3, c^j, c^j_0, c^j_1\}$. Rappelons que, dans H, il y a trois arcs indépendants de a^j_1, a^j_2 et a^j_3 à c^j, c^j_0 et c^j_1, deux arcs de c^j à 2 sommets de $\{a^j_1, a^j_2, a^j_3\}$ et pas d'arcs supplémentaires de et à des sommets appartenant à l'ensemble précédent. Sans perte de généralité, nous supposons que le seul voisin entrant de c^j dans H est a^j_1. Par construction de H, les 2 arcs précédents de c^j sont vers a^j_2 et vers a^j_3. Donc il n'y a pas de circuit avec les sommets de $\{a^j_1, a^j_2, a^j_3, c^j, c^j_0, c^j_1\}$. Cela conclut la preuve que H est sans circuit, et la preuve du Théorème 13. □

Une question intéressante est de déterminer la complexité du problème lorsqu'une longueur d'onde peut être partagée par au plus deux requêtes.

Question 1 *Quelle est la complexité du problème de décider s'il existe une reconfiguration sans interruption si une longueur d'onde peut ête partagée par au plus deux requêtes ?*

2.4.3 Modèle avec contraintes physiques

Dans cette section nous abordons le problème de la reconfiguration sous un angle différent, qui est celui de la prise en compte de contraintes physiques liées à l'établissement d'un chemin optique. Nous cherchons à optimiser le coût induit par ces contraintes sans se soucier des inévitables interruptions (nous supposons que toutes les requêtes ont des longueurs d'ondes différentes). En effet, la transmission d'un signal optique dans une fibre est sujette à de très nombreux paramètres : largeur de bande, puissance d'émission, atténuation du signal imposant l'usage d'amplificateurs tous les 50 à 80 kilomètres, décalages de phases liés à l'imperfection du laser et à la distorsion de la fibre, ou encore divers effets electro-magnétiques. Tout ceci demande des réglages extrêmement fins pour assurer une bonne qualité de transmission, mais tout est à refaire (ou adapter) lorsqu'une nouvelle longueur d'onde est utilisée dans la fibre. Ainsi, établir un nouveau chemin optique dans un réseau a un

coût (énergétique, temporel et/ou de main d'œuvre) dû aux recalibrages sur toutes les fibres empruntées par ce chemin et qui dépend de façon non linéaire des différentes longueurs d'ondes déjà présentes. De plus, ces modifications peuvent avoir des répercussions sur l'ensemble du réseau par propagation des corrections à effectuer. Voir [MCK+09] pour un exemple de prise en compte de ces effets dans le calcul du routage optique.

Le réseau physique est modélisé par un multi-digraphe $D = (V, A)$ et notons Υ l'ensemble des requêtes de connexion, avec $m = |\Upsilon|$. Étant donnés un routage initial R^{init} et un routage final R^{fin}, nous notons R_d^{init} (R_d^{fin}, respectivement) le chemin optique initial (final, respectivement) de $d \in \Upsilon$. $S \subseteq \Upsilon$ est l'ensemble des connexions déjà reroutées à une étape donnée, notée également S. C^S est la configuration du réseau à l'étape S, définie par l'ensemble $(R_d^S)_{d\in\Upsilon}$ des chemins optiques à cette étape avec $R_d^S = R_d^{fin}$ si $d \in S$ et $R_d^S = R_d^{init}$ sinon. La configuration initiale est notée $C^\emptyset$. Nous notons $l^S(e)$ la charge du lien e à l'étape S, c'est-à-dire le nombre de chemins optiques l'empruntant. D_{dep} est le graphe de dépendances dont les sommets sont Υ et il y a un arc de $d_1 \in \Upsilon$ vers $d_2 \in \Upsilon$ si $(A(R_{d_1}^{fin}) \setminus A(R_{d_1}^{init})) \cap (A(R_{d_2}^{init}) \setminus A(R_{d_2}^{fin})) \neq \emptyset$ (cette définition diffère de la définition classique énoncée dans la section 2.1.2).

2.4.3.1 Modélisation

Pour prendre en compte les contraintes physiques décrites précédemment, nous définissons le coût du reroutage de $d \in \Upsilon \setminus S$ vers la route R_d^{fin} lorsque le réseau est dans la configuration C^S par la somme (potentiellement non linéaire) de la puissance $\alpha \geq 0$ de la charge à cet instant des liens empruntés par R_d^{fin} (on note abusivement $0^\alpha = 0$, pour tout $\alpha \geq 0$), et donc du nombre de longueurs d'ondes à ajuster sur ces liens :

$$\text{coût}(d, C^S) = \sum_{e\in A(R_d^{fin})\setminus A(R_d^{init})} (l^S(e))^\alpha = \sum_{e\in A(R_d^{fin})\setminus A(R_d^{init})} |\{d' \in \Upsilon : e \in A(R_{d'}^S)\}|^\alpha.$$

Il est important de noter que le coût du déplacement d'une requête ne dépend pas de l'ordre dans lequel les requêtes précédentes ont été déplacées, mais uniquement de l'ensemble S des requêtes déjà déplacées. Par ailleurs, le cas $\alpha = 0$ est intéressant car le coût est borné par la longueur du chemin optique, i.e., par le nombre de liens du réseau sur lesquels une intervention est nécessaire, indépendemment de la charge.

Problème 3 *Étant donnés une topologie de réseau D, un ensemble Υ de requêtes, et les routages optiques initiaux et finaux, le problème de reconfiguration avec contraintes physiques consiste à trouver un ordre $\mathcal{O} = (d_1, d_2, \cdots, d_m)$ sur les requêtes afin de minimiser coût$(\mathcal{O}) = \sum_{j\leq m}$ coût(d_j, C^{S_j}), où $S_1 = \emptyset$ et $S_j = \{d_1, \cdots, d_{j-1}\}$ pour $j > 1$.*

Rappelons que nous ne considérons pas le problème du calcul du routage : les routes initiales et finales sont données comme entrées de notre problème qui consiste à ordonner le reroutage de chaque requête. Nous supposons également que le nombre

de longueurs d'ondes est suffisamment grand pour que, quelle que soit la séquence de reroutages des requêtes, aucune interruption ne soit nécessaire.

2.4.3.2 Résultats

Nous montrons tout d'abord que le problème de reconfiguration avec contraintes physiques est NP-complet même pour une instance très simple (2 nœuds et $\alpha = 0$) avant d'établir des bornes générales et un algorithme linéaire optimal pour l'anneau orienté symétrique avec $\alpha = 1$.

Théorème 14 *Si $\alpha = 0$, le problème de reconfiguration avec contraintes physiques est NP-complet pour un réseau WDM à 2 nœuds.*

Preuve : Prenons le réseau à 2 nœuds u et v et $n \geq 1$ liens uv : $A = \{a_1, \cdots, a_n\}$. Considérons un ensemble de m requêtes de u à v ayant chacune un lien pour route initiale et un lien différent pour route finale. Sans perte de généralité, nous supposons que chaque lien est la route finale d'au moins une requête.

Comme $\alpha = 0$, remarquons que le reroutage d'une requête ne coûte rien si elle est déplacée sur un lien vide, et 1 sinon. Plus précisément, étant donné un lien $a \in A$ qui est la route finale de $c(a) \geq 1$ requêtes, la contribution de ce lien au coût total de la reconfiguration est de $c(a) - 1$ ou $c(a)$ selon que ce lien est vide ou non lorsque la première requête l'ayant pour route finale est déplacée. En d'autres termes, quel que soit l'ordre $\mathcal{O}$ adopté, $m - n = \sum_{i \leq n}(c(a_i) - 1) \leq \text{coût}(\mathcal{O}) \leq \sum_{i \leq n} c(a_i) = m$.

Considérons le graphe auxiliaire G_{aux} dont A est l'ensemble des sommets et il y a autant d'arcs $a_i a_j$ que de requêtes dont la route initiale est a_i et la route finale a_j. On peut montrer que le coût optimal de la reconfiguration vaut $m - n + |MFVS(G_{aux})|$. Rappelons que le problème de déterminer $MFVS(G_{aux})$, un minimum feedback vertex set de G_{aux}, est NP-complet [Kan92].

Pour conclure la preuve, il est trivial de montrer que pour tout graphe orienté G, il existe une instance de notre problème pour laquelle $G_{aux} = G$. □

Bien que le problème soit NP-complet en général, il existe des instances où il peut être résolu efficacement. Entre autres, nous proposons un algorithme linéaire optimal pour un anneau orienté symétrique si $\alpha = 1$. Nous commençons avec des bornes générales.

Notons $I(a) = |\{d \in \Upsilon \ : \ a \in A(R_d^{init}) \setminus A(R_d^{fin})\}|$, $F(a) = |\{d \in \Upsilon \ : \ a \in A(R_d^{fin}) \setminus A(R_d^{init})\}|$ et $P(a) = |\{d \in \Upsilon \ : \ a \in A(R_d^{fin}) \cap A(R_d^{init})\}|$ le nombre de requêtes utilisant le lien a uniquement dans leurs routes initiales, finales, de façon permanente, respectivement. Enfin, soit $A' \subseteq A$, l'ensemble des liens a tels que $F(a) > 0$. Les bornes suivantes sont très simples à obtenir (nous notons abusivement $\sum_{i=x}^{y} i^\alpha = 0$ si $y < x$).

Lemme 14 *Pour tout $\alpha \geq 0$ et pour tout ordre $\mathcal{O}$ de déplacement des requêtes,*

$$\sum_{a \in A'} \sum_{i=P(a)}^{P(a)+F(a)-1} i^\alpha \leq \text{coût}(\mathcal{O}) \leq \sum_{a \in A'} \sum_{i=P(a)+I(a)}^{P(a)+I(a)+F(a)-1} i^\alpha.$$

En particulier, si $I(a) = 0$ pour tout $a \in A'$, le coût optimal est atteint pour n'importe quel ordre de déplacement des requêtes. Plus généralement,

Théorème 15 *Pour tout $\alpha \geq 0$, si le graphe de dépendances D_{dep} de l'instance de reconfiguration est acyclique, alors tout ordre d'effeuillage de D_{dep} est optimal de coût $\sum_{a \in A'} \sum_{i=P(a)}^{P(a)+F(a)-1} i^{\alpha}$.*

Soit $\mathcal{O} = (d_1, \cdots, d_m)$ un ordre sur les requêtes et $\mathcal{O}'$ l'ordre obtenu de $\mathcal{O}$ en inversant les i^e et $(i+1)^e$ requêtes. Insistons sur le fait que la différence de coût des deux ordres dépend uniquement de $S_i = \{d_1, \cdots, d_{i-1}\}$:

$$\text{coût}(\mathcal{O}) - \text{coût}(\mathcal{O}') = \text{coût}(d_i, C^{S_i}) + \text{coût}(d_{i+1}, C^{S_{i+1}}) - \text{coût}(d_{i+1}, C^{S_i}) - \text{coût}(d_i, C^{S_i \cup \{d_{i+1}\}}).$$

Théorème 16 *Soit un anneau orienté symétrique et un ensemble de demandes à déplacer $\{d_1, \cdots, d_m\}$ rangées par ordre décroissant des longueurs de leurs routes initiales. Cet ordre est optimal pour la reconfiguration.*

Preuve : Considérons un ordre $\mathcal{O}$ quelconque et l'ordre $\mathcal{O}'$ obtenu de $\mathcal{O}$ en inversant deux demandes consécutives d et d'. L'argument clé (dû à la linéarité de la fonction) est que la différence de coût entre les deux ordres ne dépend que de d et d' (inutile de connaître l'ensemble des requêtes précédent d et d'). Plus précisément, si d et d' n'ont pas le même sens dans le routage initial ($\text{coût}(\mathcal{O}) = \text{coût}(\mathcal{O}')$ sinon), $\text{coût}(\mathcal{O}) - \text{coût}(\mathcal{O}') = |A(R_{d'}^{init}) \cap A(R_d^{fin})| - |A(R_d^{init}) \cap A(R_{d'}^{fin})|$ ce qui, dans un anneau, vaut $|A(R_{d'}^{init})| - |A(R_d^{init})|$. Permuter successivement les requêtes dans $\mathcal{O}$ jusqu'à obtenir un ordre décroissant sur les longueurs des routes initiales n'augmente donc pas le coût. Le coût obtenu est donc optimal. □

Pour conclure cette section, nous avons également proposé différentes heuristiques pour ce problème et nous les avons évaluées via simulations [BCM+11].

2.5 Conclusion et perspectives

Le problème de reconfiguration des requêtes de connexion dans les réseaux optiques soulève de nombreuses questions d'ordres très variés.

D'un point de vue de la modélisation, nous avons montré que la plupart des problèmes d'optimisation de paramètres importants liés aux interruptions des requêtes, se traduisent par des problèmes de graphes : calcul de l'indice de traitement ou de l'indice de transmission du digraphe associé à l'instance (minimisation du nombre maximum de requêtes simultanément interrompues ou minimisation du nombre total de requêtes interrompues). Nous avons proposé un algorithme distribué pour le calcul de l'indice de traitement dans les arbres orientés symétriques. Il peut être adapté au calcul de l'indice d'échappement sommet, entre autres paramètres. Nous avons ensuite introduit des paramètres de compromis prenant en compte simultanément ces deux métriques. De plus, nous avons montré que le problème de reconfiguration avec un sous-ensemble de requêtes prioritaires ne pouvant pas être interrompues,

peut se ramener en temps polynomial au problème initial. Nous avons également considéré un modèle avec partage de la longueur d'onde et montré sa grande difficulté : le problème de décider s'il existe une reconfiguration sans aucune interruption est NP-complet si chaque longueur d'onde peut être partagée par trois connexions. Enfin, nous avons proposé une modélisation prenant en compte des contraintes physiques. Encore une fois, nous avons mis en exergue sa grande difficulté : le problème de minimiser le coût de la reconfiguration est NP-complet même pour un réseau physique composé de 2 nœuds.

D'un point de vue théorique, de nombreuses questions restent actuellement sans réponse. Pouvons-nous adapter notre algorithme calculant l'indice de traitement des arbres orientés symétriques pour le calcul de ce paramètre dans des classes de graphes proches de celle de l'arbre ? Pouvons-nous notamment en déduire un algorithme ayant une complexité raisonnable (biens moins que le $O(n^{11})$ de [BK96]) pour la classe des graphes planaires extérieurs biconnexes ? Est-ce que le rapport $\frac{mfvs_{pn}(D)}{mfvs(D)}$ est plus petit que 3 pour tout digraphe orienté symétrique (Conjecture 1) ? Quelle est la complexité du problème de décider s'il existe une reconfiguration sans interruption si une longueur d'onde peut ête partagée par au plus deux requêtes (Question 1) ?

Il reste de nombreuses pistes à explorer concernant la modélisation du problème de la reconfiguration du routage. Entre autres questions, comment prendre en compte la durée des interruptions ? Devons-nous minimiser la somme des durées ou la durée la plus grande ou un compromis entre les deux ? Il est également pertinent de minimiser le nombre d'étapes de la reconfiguration. Il faut alors définir précisément la notion d'étape. Voir [Cou10] pour plus de détails sur la modélisation avec prise en compte du temps.

D'un point de vue pratique, nous avons proposé dans [CHM+09a] des heuristiques pour le calcul de stratégies de traitement. Nous poursuivons les comparaisons en effectuant notamment des comparaisons avec les heuristiques de [JS03, SPa10].

Chapitre 3

Routage efficace en énergie

Sommaire

3.1 Introduction

Dans le cadre de l'ANR Jeunes Chercheuses Jeunes Chercheurs DIMAGREEN[1], nous nous sommes intéressés à la consommation d'énergie dans les réseaux et principalement à celle liée au routage des demandes. L'objectif de ce chapitre est d'étudier comment le routage influe sur la consommation d'énergie et de proposer des solutions économes en énergie. Ce travail a été réalisé avec Frédéric Giroire, Joanna Moulierac et Brice Onfroy [GMMO10a, GMMO10b, GMM11].

[1] http://www-sop.inria.fr/teams/mascotte/Contrats/DIMAGREEN

3.1.1 Motivations et état de l'art

Les technologies de l'information et de la communication (TIC) sont responsables à elles seules de 2% à 10% (selon les estimations) de la consommation mondiale [CLM09]. Dans ce chapitre, nous nous intéressons à la consommation liée aux réseaux. Il est estimé que les concentrateurs (*hubs*), commutateurs (*switches*) et routeurs consomment 6 TWh par an aux USA [NC05]. Un certain nombre d'études de la consommation d'énergie des réseaux ont été effectuées ces dernières années [MSBR08, MSBR09, CSB+08]. Les auteurs mettent en exergue le fait que la consommation des équipements est indépendante de leurs charges. En particulier dans [CSB+08, MSBR09], les auteurs se sont intéressés à la consommation d'énergie des routeurs. Par exemple, la consommation du Cisco 12000 lorsque la charge est de 75% est seulement 2% plus grande que sa consommation en état d'inactivité (770W contre 755W). Dans [MSBR09], des expérimentations montrent que la consommation d'énergie dépend du nombre de ports activés. Ainsi éteindre certains ports non utilisés d'une line card réduit la consommation d'énergie de l'équipement. Les valeurs obtenues lors des expérimentations montrent que la consommation d'une line card 4-port Gigabit ethernet (100W) représente approximativement un quart de la consommation globale du système (430W).

Ainsi, la consommation dépend principalement du nombre d'équipements du réseau allumés et donc du routage des demandes. Dans ce contexte, il s'agit alors de trouver un routage des demandes qui minimise le nombre d'équipements utilisés.

Dans [CSB+08], les auteurs proposent un programme linéaire pour résoudre ce problème. La fonction objectif est la somme pondérée du nombre de plateformes et d'interfaces. Ils montrent expérimentalement la quantité d'énergie qui peut être économisée dans différents réseaux avec ce modèle. Dans [IOR+10], les auteurs s'intéressent au reroutage sur différentes couches dans les réseaux IP-sur-WDM pour économiser de l'énergie. Dans [PVC+09], l'impact de la technologie sur l'efficacité du routage en termes d'énergie est étudié. Dans [NIRW08, GS03, CMN09a], différentes techniques sont présentées comme par exemple adapter le taux d'envoi des messages en fonction du trafic dans les LAN (*Local Area Networks*). Dans [CNR10], les auteurs proposent une modulation de la configuration radio dans les réseaux sans-fil à large bande pour réduire la consommation.

Ces problèmes étant très difficiles, nous considérons une architecture simplifiée dans laquelle une connexion entre deux routeurs est représentée par un lien reliant deux interfaces.

Le réseau est modélisé par un graphe arête-pondéré non-orienté $G = (V, E, c)$, $c : E \to \mathbb{R}_+^*$, avec $c(e) \geq 0$ représentant la capacité de l'arête $e \in E$. Nous notons $n = |V|$ et $m = |E|$. L'ensemble des demandes est noté $\mathcal{D} = \{\mathcal{D}_{st} \geq 0; (s,t) \in V \times V, s \neq t\}$ avec $\mathcal{D}_{st}$ le volume de trafic de s à t. Une demande $\mathcal{D}_{st}$ doit être routée via un *unique chemin* de s à t. En d'autres termes, les demandes sont insécables (*unsplittable*). Un routage valide des demandes est un ensemble de chemins dans G pour chacune des demandes $\mathcal{D}_{st} \in \mathcal{D}$, tel que, pour chaque arête $e \in E$, le volume total de trafic passant par e n'excède pas sa capacité $c(e)$. Comme dans [CLBG00],

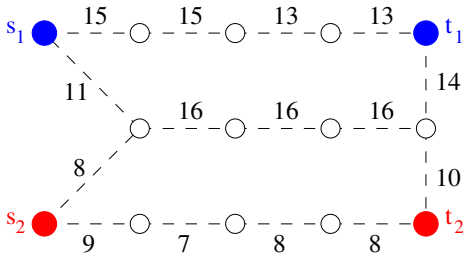


FIG. 3.1 – Instance n'admettant pas de solution pour le PROBLÈME DE ROUTAGE (et donc pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM) avec $\mathcal{D}_{s_1t_1} = 10$ et $\mathcal{D}_{s_2t_2} = 10$. Les entiers sur les arêtes représentent leurs capacités respectives.

pour toute arête $\{uv\} \in E$, nous ne faisons pas de distinction entre le trafic passant de u et vers v et le trafic passant de v vers u. Un problème de décision classique est le PROBLÈME DE ROUTAGE qui consite à déterminer si un tel routage existe ou non.

Problème 4 (problème de routage) *Etant donnés un graphe pondéré $G = (V, E, c)$ et un ensemble de demandes $\mathcal{D}$, le* PROBLÈME DE ROUTAGE *consiste à décider s'il existe un routage valide des demandes de $\mathcal{D}$ dans G.*

Considérons le graphe de la figure 3.1 avec les deux demandes $\mathcal{D}_{s_1t_1} = 10$ et $\mathcal{D}_{s_2t_2} = 10$. Les entiers sur les arêtes représentent leurs capacités respectives. Il n'y a pas de solution pour le PROBLÈME DE ROUTAGE. En effet, il est impossible de router $\mathcal{D}_{s_2t_2} = 10$ car le sommet s_2 a deux arêtes adjacentes de capacités 8 et 9, respectivement. Rappelons que chaque demande doit être routée via un unique chemin.

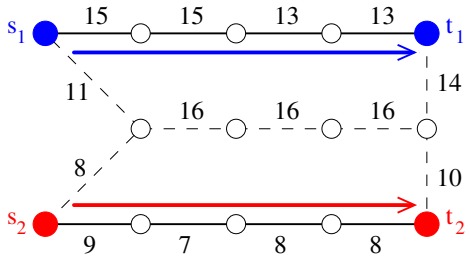
Le PROBLÈME DE ROUTAGE est bien connu pour être NP-complet même pour deux demandes [EIS75].

3.1.2 Minimiser le nombre d'arêtes lorsque les capacités et les demandes sont fixées

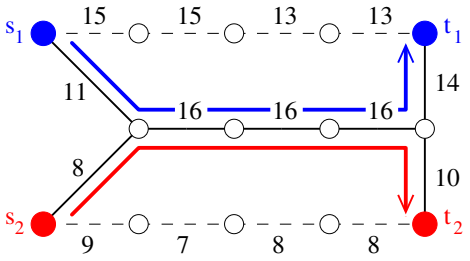
Dans le but d'économiser de l'énergie, il est possible d'éteindre deux interfaces qui sont aux deux extrémités d'un lien reliant deux routeurs. Dans notre modélisation, l'objectif est alors de *trouver un sous-graphe minimum en nombre d'arêtes permettant un routage valide des demandes.* Formellement.

Problème 5 (problème de routage arêtes minimum) *Etant donnés un graphe pondéré $G = (V, E, c)$ et un ensemble de demandes $\mathcal{D}$, le* PROBLÈME DE ROUTAGE ARÊTES MINIMUM *consiste à trouver un ensemble $E^* \subseteq E$ de cardinalité minimum tel qu'il existe un routage valide des demandes $\mathcal{D}$ dans $G^* = (V, E^*, c^*)$, $c^* : E^* \to \mathbb{R}_+^*$, avec $c^*(e) = c(e)$, pour tout $e \in E^*$.*

Le PROBLÈME DE ROUTAGE ARÊTES MINIMUM est un cas particulier de problèmes classiques d'optimisation dans les réseaux : problème de routage de coût minimum [YJ71], problème de flot concave minimum [GP90], problème de flot minimum



(a) $\mathcal{D}_{s_1t_1} = 12$ et $\mathcal{D}_{s_2t_2} = 5 \rightarrow |E^*| = 8$. La solution optimale donne deux plus courts chemins.



(b) $\mathcal{D}_{s_1t_1} = 10$ et $\mathcal{D}_{s_2t_2} = 5 \rightarrow |E^*| = 7$. Les deux routes de la solution optimale ne sont pas des plus courts chemins.

FIG. 3.2 – Deux solutions optimales pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM pour deux instances légèrement différentes. Les entiers sur les arêtes représentent leurs capacités respectives.

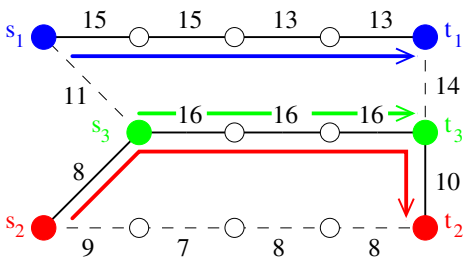
avec fonctions de coût [GKM99]. En recherche opérationnelle, le problème peut être vu comme un cas particulier du *fixed charge transportation problem* [Spi64, Dan63], où le coût du flot unitaire sur une arête est zéro. Notons que ce problème est NP-difficile [GP90].

Nous présentons à présent des instances simples ainsi que les solutions optimales associées pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM. Nous considérons le graphe de la figure 3.1 où les entiers représentent les capacités des arêtes.

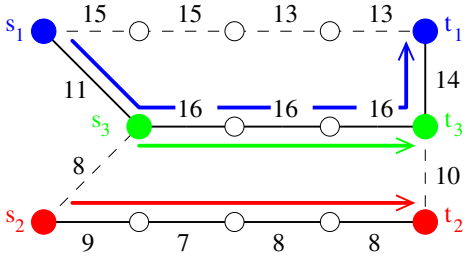
Si les deux demandes sont $\mathcal{D}_{s_1t_1} = 10$ et $\mathcal{D}_{s_2t_2} = 10$, alors il n'y a pas de solution pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM (figure 3.1).

Dans la figure 3.2(a), il y a deux demandes $\mathcal{D}_{s_1t_1} = 12$ et $\mathcal{D}_{s_2t_2} = 5$. La solution optimale E^* pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM est composée de $|E^*| = 8$ et les deux demandes sont routées via leurs plus courts chemins respectifs. Dans la figure 3.2(b), il y a deux demandes $\mathcal{D}_{s_1t_1} = 10$ et $\mathcal{D}_{s_2t_2} = 5$. La solution optimale E^* pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM est composée de $|E^*| = 7$ arêtes. Remarquons que le chemin de s_i à t_i dans $G^* = (V, E^*)$ est composé de 5 arêtes, alors que le plus court chemin dans G est composé de 4 arêtes, pour $i \in \{1, 2\}$. En effet, les deux plus courts chemins sont arête-disjoints alors que dans la solution E^*, les deux chemins partagent 3 arêtes. Cet exemple très simple montre que le routage par plus courts chemins ne donnent pas (nécessairement) une solution optimale.

Dans la figure 3.3, il y a trois demandes $\mathcal{D}_{s_1t_1} = 10$, $\mathcal{D}_{s_2t_2} = 5$ et $\mathcal{D}_{s_3t_3} = 2$. A cause des trois arêtes de capacité 16, seulement deux demandes peuvent être routées via ces trois arêtes. Il y ainsi deux solutions optimales avec $|E^*| = 9$ représentées dans la figure 3.3(a) et dans la figure 3.3(b).



(a) $\mathcal{D}_{s_1t_1}$ et $\mathcal{D}_{s_3t_3}$ sont routées via des plus courts chemins.



(b) $\mathcal{D}_{s_2t_2}$ et $\mathcal{D}_{s_3t_3}$ sont routées via des plus courts chemins.

FIG. 3.3 – Deux solutions optimales différentes pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM pour une même instance avec $\mathcal{D}_{s_1t_1} = 10$, $\mathcal{D}_{s_2t_2} = 5$ et $\mathcal{D}_{s_3t_3} = 2$. Les entiers sur les arêtes représentent leurs capacités respectives.

3.1.3 Minimiser le rapport capacité sur demande lorsque le nombre d'arêtes est fixé

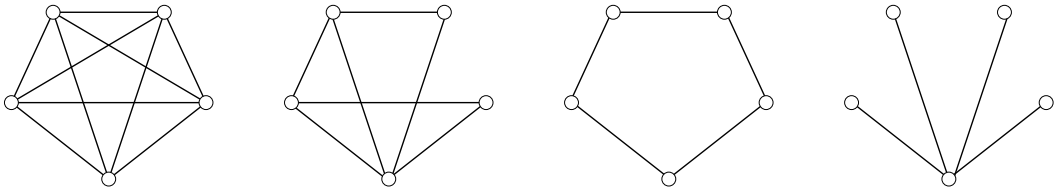
Nous définissons dans cette section un problème orthogonal au PROBLÈME DE ROUTAGE ARÊTES MINIMUM. Nous supposons que toutes les capacités des arêtes sont identiques et égales à une constante c et les demandes sont de type *all-to-all* ou *one-to-all*, chacune d'entre elles ayant un volume constant κ. Nous définissons le rapport λ entre ces deux valeurs.

Définition 26 (rapport capacité sur demande) *Le rapport capacité sur demande λ exprime le lien entre la capacité des arêtes et le volume des demandes : $\lambda = c/\kappa$.*

Nous définissons à présent le problème orthogonal au PROBLÈME DE ROUTAGE ARÊTES MINIMUM consistant à déterminer le plus petit rapport λ qui permette un routage valide des demandes lorsque les demandes sont de type *all-to-all* et que le nombre d'arêtes est contraint.

Problème 6 (problème de routage rapport minimum (*all-to-all*)) *Etant donnés un graphe connexe pondéré $G = (V, E, c)$ avec $c(e) = c$ pour tout $e \in E$, un ensemble de demandes $\mathcal{D} = \{\mathcal{D}_{st} = \kappa; (s,t) \in V \times V, s \neq t\}$ (*all-to-all*) et un entier q, $n - 1 \leq q \leq m$, le* PROBLÈME DE ROUTAGE RAPPORT MINIMUM *consiste à déterminer le rapport λ_q qui est le plus petit $\lambda = c/\kappa$ tel qu'il existe un ensemble $E^q \subseteq E$ de cardinalité $|E^q| \leq q$ tel qu'il existe un routage valide des demandes dans le graphe $G^q = (V, E^q, c)$.*

Soit une instance du PROBLÈME DE ROUTAGE RAPPORT MINIMUM définie par un graphe $G = (V, E, c)$, des demandes de type *all-to-all* de volume κ et un entier q. Soit λ_q la solution. Nous définissons une instance du PROBLÈME DE ROUTAGE ARÊTES MINIMUM par le graphe $G = (V, E, c)$ avec $c(e) = c$ pour tout $e \in E$ et des



(a) $\lambda_{m=10} = 2$. (b) $\lambda_7 = 4$. (c) $\lambda_5 = 6$. (d) $\lambda_{n-1=4} = 8$.

FIG. 3.4 – $\lambda_{m=10} = 2$, $\lambda_7 = 4$, $\lambda_5 = 6$ et $\lambda_{n-1=4} = 8$: quatre valeurs optimales pour le PROBLÈME DE ROUTAGE RAPPORT MINIMUM pour le graphe complet K_5 composé de 5 sommets et de 10 arêtes et des demandes de type *all-to-all*.

demandes de type *all-to-all* de volume constant κ vérifiant $\lambda_q = c/\kappa$. Nous déduisons alors qu'une solution E^* pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM vérifie $|E^*| \leq q$. Nous n'avons pas nécessairement l'égalité car si $q = |E|$ par exemple, un routage optimal peut ne pas utiliser toutes les arêtes du graphe.

Nous nous intéressons particulièrement aux deux valeurs extrêmes de λ : λ_m et λ_{n-1}. Le rapport λ_m représente la plus petite valeur du rapport capacité sur demande λ pour laquelle le PROBLÈME DE ROUTAGE ARÊTES MINIMUM (et le PROBLÈME DE ROUTAGE) admet une solution. Il est possible de relier la valeur λ_m à la notion de charge du graphe et à la notion de *edge-forwarding index* introduite dans [HMS89].

Définition 27 (charge du graphe, *edge-forwarding index*) *Etant donnés un graphe connexe pondéré $G = (V, E, c)$ avec $c(e) = c$ pour tout $e \in E$, un ensemble de demandes $\mathcal{D} = \{\mathcal{D}_{st} = 1; (s,t) \in V \times V, s \neq t\}$ (*all-to-all*), le* edge-forwarding index *est le plus petit c tel qu'il existe un routage valide des demandes dans G.*

Comme le problème de déterminer la charge du graphe est NP-complet, alors le problème de déterminer λ_m l'est aussi.

Le rapport λ_{n-1} représente la plus petite valeur du rapport λ pour laquelle il existe un routage des demandes via un arbre couvrant de G. En effet, les demandes étant de type *all-to-all*, $|V| - 1$ arêtes sont nécessaires. En d'autres termes, λ_{n-1}/κ représente la charge minimum parmi l'ensemble des charges de tous les arbres possibles admettant un routage valide des demandes.

Considérons le graphe complet K_5 représenté dans la figure 3.4 et des demandes de type *all-to-all*. Pour cette instance, nous avons $\lambda_{m=10} = 2$ car chaque demande est routée via un plus court chemin correspondant à une arête (figure 3.4(a)). Nous pouvons facilement vérifier que $\lambda_7 = 4$ (figure 3.4(b)). De manière orthogonale, le sous-graphe de la figure 3.4(b) représente une solution pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM lorsque $\lambda = 4$. De manière analogue, nous avons $\lambda_5 = 6$ (figure 3.4(c)). Le graphe de la figure 3.4(c) est une solution pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM si $\lambda = 6$. Enfin en prenant un arbre formant une étoile (figure 3.4(d)), nous obtenons $\lambda_{n-1=4} = 8$. Si nous avions pris un chemin par

exemple, nous aurions obtenu $\lambda = 12$. Si $\lambda \geq 8$, alors le graphe de la figure 3.4(d) est une solution pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM.

Nous définissons le problème analogue lorsque les demandes sont de type *one-to-all*.

Problème 7 (problème de routage rapport minimum (*one-to-all*)) *Etant donnés un graphe connexe pondéré $G = (V, E, c)$ avec $c(e) = c$ pour tout $e \in E$, un ensemble de demandes $\mathcal{D} = \{\mathcal{D}_{rt} = \kappa; t \in V, t \neq r\}$ avec $r \in V$ un sommet fixé (*one-to-all*) et un entier q, $n - 1 \leq q \leq m$, le* PROBLÈME DE ROUTAGE RAPPORT MINIMUM *consiste à déterminer le rapport λ_q qui est le plus petit $\lambda = c/\kappa$ tel qu'il existe un ensemble $E^q \subseteq E$ de cardinalité $|E^q| \leq q$ tel qu'il existe un routage valide des demandes dans le graphe $G^q = (V, E^q, c)$.*

Dans la suite et pour tout type de demandes, nous utiliserons le terme charge d'une arête e comme étant le nombre de messages passant par e. De manière analogue, nous utiliserons le terme charge du graphe comme la valeur de la charge de l'arête la plus chargée.

3.1.4 Contributions

Dans la section 3.2, nous nous intéressons à la complexité des problèmes définis précédemment et notamment à leurs inapproximabilités. Nous mettons tout d'abord en exergue des résultats négatifs concernant des heuristiques gloutonnes (section 3.2.1). Nous montrons que des heuristiques gloutonnes basées sur les plus courts chemins ou sur le nombre d'arêtes ajoutées à la solution courante peuvent donner un rapport non borné entre les cardinalités de la solution heuristique et d'une solution optimale. Cela reste vrai même si le meilleur ordre sur les demandes est donné en entrée. Plus généralement, nous prouvons dans la section 3.2.2 que le PROBLÈME DE ROUTAGE ARÊTES MINIMUM n'est pas dans APX, c'est-à-dire qu'il n'existe pas d'algorithme polynomial d'approximation à un facteur constant près, à moins que P=NP. Cela reste vrai même pour des instances très simples : avec seulement deux demandes ou si les capacités des arêtes sont constantes. Dans la section 3.2.3, nous étudions la complexité du PROBLÈME DE ROUTAGE RAPPORT MINIMUM pour des demandes de type *one-to-all*. Nous montrons que le problème de déterminer λ_{n-1} est NP-difficile. Nous montrons de plus que déterminer λ_{n-1} n'est pas dans APX si nous ajoutons une contrainte supplémentaire sur le degré maximum de certains sommets dans l'arbre solution. Cela reste vrai même si le degré maximum dans l'arbre est un $\Omega(n^{\frac{k}{k+2}})$ avec n le nombre de sommets et k n'importe quelle constante.

Dans la section 3.3, nous prouvons des bornes théoriques et des valeurs exactes des paramètres définis dans la section 3.1.2 et dans la section 3.1.3 pour différentes classes de graphes. Nous étudions prémièrement dans la section 3.3.1 les graphes généraux. Nous débutons par décrire des bornes classiques concernant les capacités et les volumes des demandes permettant d'assurer un routage valide des demandes.

Dans le reste de la section, nous supposons que les demandes sont de type *all-to-all*. Nous prouvons une borne inférieure pour λ_{n-1} pour un graphe quelconque en fonction de son degré maximum. Dans la section 3.3.2, nous nous intéressons au cas particulier du graphe complet pour lequel nous avons $\lambda_m = 2$ et $\lambda_{n-1} = 2(n-1)$. L'écart entre ces deux valeurs étant important, nous prouvons une borne inférieure sur le nombre d'arêtes utilisées en fonction de λ. Enfin, nous nous intéressons dans la section 3.3.3 aux grilles. Nous prouvons une formule close pour λ_{n-1} et nous donnons ensuite une borne inférieure pour λ_m. Nous donnons également une borne supérieure (construction d'un graphe atteignant cette borne) et une borne inférieure pour les valeurs de λ_q intermédiaires.

Comme les problèmes étudiés dans ce chapitre sont NP-difficiles et parfois même pas dans APX, nous proposons dans la section 3.4 des heuristiques pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM et le PROBLÈME DE ROUTAGE RAPPORT MINIMUM (section 3.4.1). Nous comparons les résultats obtenus par les heuristiques avec le programme linéaire entier (lorsque cela est possible), les valeurs exactes et les bornes obtenues dans la section 3.3 pour la topologie de grille (section 3.4.2). Dans la section 3.4.3, nous faisons de même pour des topologies réelles : nous étudions le *gain en termes d'énergie* pour un ensemble de réseaux backbone existants. Nous montrons qu'*au moins un tiers des interfaces du réseau* peut être économisé pour un ensemble de demandes *all-to-all*. Nous discutons également l'impact de ces solutions efficaces en énergie sur la *longueur des routes* et sur la *tolérance aux pannes*.

3.2 Résultats d'inapproximabilité

Dans cette section nous montrons premièrement que le routage par plus courts chemins peut donner une solution arbitrairement mauvaise par rapport à une solution optimale (Lemme 15). De plus, un algorithme glouton routant les demandes via un plus court chemin ou un chemin qui minimise le nombre d'arêtes ajoutées à la solution courante peut également donner une solution arbitrairement mauvaise quel que soit l'ordre dans lequel les demandes sont routées (Corollaire 7). Plus généralement, nous prouvons ensuite que le PROBLÈME DE ROUTAGE ARÊTES MINIMUM n'est pas dans APX. Cela reste vrai même avec deux instances très particulières (Théorème 17 et Théorème 18). Enfin, nous prouvons dans le Théorème 20 que le problème de déterminer λ_{n-1} n'est pas dans APX si nous ajoutons une contrainte supplémentaire sur les degrés des sommets dans l'arbre.

3.2.1 Algorithmes gloutons

Dans cette section, nous montrons qu'un algorithme glouton basé sur les plus courts chemins ou sur le nombre minimum d'arêtes à ajouter à chaque étape peut donner un nombre d'arêtes arbitrairement grand comparé à $|E^*|$. Cela reste vrai même si le meilleur ordre pour effectuer le routage glouton nous est donné en entrée.

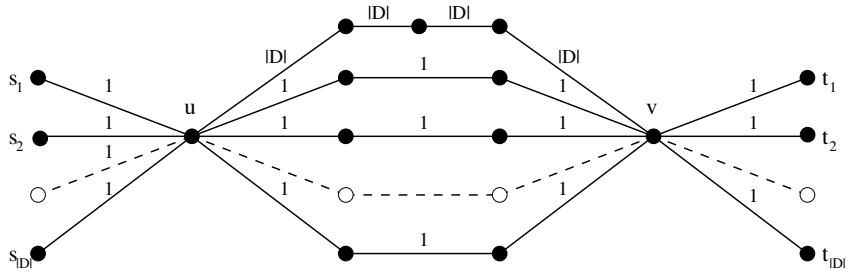


FIG. 3.5 – Graphe $G = (V, E, c)$ pour $x = 3$ utilisé dans la preuve du Lemme 15 et dans le Corollaire 7. Entre u et v, il y a $|\mathcal{D}|+1$ chemins élémentaires : 1 est composé de $x+1 = 4$ arêtes de capacité $|D|$ et $|\mathcal{D}|$ sont composés de $x = 3$ arêtes de capacité 1. Il y a $|\mathcal{D}|$ demandes $\mathcal{D}_{s_1t_1}, \mathcal{D}_{s_2t_2}, \ldots, \mathcal{D}_{s_{|\mathcal{D}|}t_{|\mathcal{D}|}}$, chacune de volume 1.

Définition 28 (problème de routage par plus court chemin (PCC)) *Etant donnés un graphe pondéré $G = (V, E, c)$ et un ensemble de demandes $\mathcal{D}$, le* PROBLÈME DE ROUTAGE PAR PLUS COURT CHEMIN *consiste à trouver un ensemble $E^{PCC} \subseteq E$ de cardinalité minimum tel qu'il existe un routage valide des demandes $\mathcal{D}$ dans $G^{PCC} = (V, E^{PCC}, c^{PCC})$, $c^{PCC} : E^{PCC} \to \mathbb{R}^*_+$, avec $c^{PCC}(e) = c(e)$, pour tout $e \in E^{PCC}$, de telle sorte que chaque demande soit routée via un plus court chemin de G.*

Le Lemme 15 prouve que le rapport entre $|E^{PCC}|$ et $|E^*|$ n'est pas borné.

Lemme 15 *$\forall k > 1$, il existe un graphe pondéré $G = (V, E, c)$ et un ensemble de demandes $\mathcal{D}$ tels que $|E^{PCC}| > k|E^*|$.*

Preuve : Considérons le graphe pondéré $G = (V, E, c)$ décrit dans la figure 3.5. Les entiers sur les arêtes représentent les différentes capacités $c(e)$ pour chacune des arêtes $e \in E$. Soient $|\mathcal{D}|$ et x deux entiers. Il y a $|\mathcal{D}|$ sommets $s_1, \ldots, s_{|\mathcal{D}|}$ reliés au sommet u par des arêtes de capacité 1. Il y a $|\mathcal{D}|$ sommets $t_1, \ldots, t_{|\mathcal{D}|}$ reliés au sommet v par des arêtes de capacité 1. Entre u et v, il y a $|\mathcal{D}|$ chemins arête-disjoints composés de x arêtes chacun (toutes de capacité 1). Il y a également un chemin entre u et v composé de $x+1$ arêtes chacune de capacité $|D|$. Dans la figure 3.5, G est représenté pour $x = 3$. Les $|\mathcal{D}|$ demandes sont $\mathcal{D}_{s_1t_1}, \mathcal{D}_{s_2t_2}, \ldots, \mathcal{D}_{s_{|\mathcal{D}|}t_{|\mathcal{D}|}}$, chacune de volume 1. $\forall i \in \{1, \ldots, |\mathcal{D}|\}$, la longueur d'un plus court chemin de s_i à t_i est $x+2$. Si chacune des demandes est routée via un plus court chemin, alors il est facile d'observer que le routage utilise $|E^{PCC}| = (x+2)|\mathcal{D}|$ arêtes car toutes les arêtes de tout plus court chemin ont capacité 1. En effet, toute paire de demandes vérifie que ces dernières sont routées via deux (plus courts) chemins arête-disjoints. Cependant, il y a un routage optimal E^* pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM tel que $|E^*| = 2|\mathcal{D}| + x + 1$: chaque demande est routée via les $x+1$ arêtes de capacité $|D|$. Remarquons que pour ce routage optimal, aucune des demandes n'est routée via un plus court chemin.

Pour toute constante $k > 1$, si nous choisissons $x > \frac{2|\mathcal{D}|(k-1)+k}{|\mathcal{D}|-k}$ et en prenant $|\mathcal{D}| > k$, nous obtenons l'inégalité désirée. □

Soit $\mathcal{O}(\mathcal{D})$ un ordre sur les demandes. L'ALGORITHME GLOUTON PAR PLUS COURT CHEMIN consiste à déterminer séquentiellement selon $\mathcal{O}(\mathcal{D})$ une route pour chacune des demandes avec la contrainte que cette dernière est un plus court chemin. L'ALGORITHME GLOUTON PAR MINIMISATION DU NOMBRE DE NOUVELLES ARÊTES consiste à déterminer séquentiellement selon $\mathcal{O}(\mathcal{D})$ une route pour chacune des demandes en minimisant le nombre d'arêtes ajoutées à la solution courante. Nous notons $E(\mathcal{O}(\mathcal{D}))$ l'ensemble d'arêtes retourné par une de ces deux heuristiques. Nous déduisons du Lemme 15 que le rapport entre $|E^*|$ et $|E(\mathcal{O}(\mathcal{D}))|$ n'est pas borné :

Corollaire 7 *$\forall k > 1$, il existe un graphe pondéré $G = (V, E, c)$ et un ensemble de demandes $\mathcal{D}$ tels que pour tout ordre $\mathcal{O}(\mathcal{D})$ sur les demandes, nous avons $|E(\mathcal{O}(\mathcal{D}))| > k|E^*|$.*

Pour montrer ce résultat, nous utilisons l'instance de la preuve du Lemme 15. En effet, les arêtes de capacité $|D|$ ne sont pas choisies par les deux heuristiques. Dans la prochaine section, nous montrons un résultat négatif plus général pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM.

3.2.2 Le problème de routage arêtes minimum n'est pas dans APX

Nous prouvons que le PROBLÈME DE ROUTAGE ARÊTES MINIMUM n'est pas dans APX même pour deux classes d'instances particulières (Théorème 17 et Théorème 18). En d'autres termes, pour tout $k \geq 1$, il n'existe pas d'algorithme polynomial garantissant $|E'| \leq k|E^*|$ où E' est la solution retournée par cet algorithme, à moins que P=NP.

Théorème 17 *Le* PROBLÈME DE ROUTAGE ARÊTES MINIMUM *n'est pas dans APX même avec $|\mathcal{D}| = 2$ demandes.*

Preuve : Soit $G = (V, E, c)$ un graphe pondéré. Soient $\mathcal{D}_{s_1t_1} > 0$ et $\mathcal{D}_{s_2t_2} > 0$ les deux seules demandes. Soit $k \geq 1$. Nous construisons $G' = (V', E')$ comme suit : nous partons d'une copie de G ; nous ajoutons un chemin P_1 entre s_1 et t_1 composé de $x > k|E|$ arêtes et un chemin P_2 entre s_2 et t_2 composé également de $x > k|E|$ arêtes. Chaque arête de P_1 et P_2 a capacité $\mathcal{D}_{s_1t_1}+\mathcal{D}_{s_2t_2}$. Nous considérons les mêmes demandes $\mathcal{D}_{s_1t_1} > 0$ et $\mathcal{D}_{s_2t_2} > 0$. La figure 3.6 représente cette construction.

Supposons maintenant qu'il existe une constante $k \geq 1$ telle qu'il existe un algorithme polynomial trouvant un sous-ensemble $E^k \subseteq E$, garantissant l'existence d'un routage des demandes utilisant uniquement les arêtes de E^k et assurant $|E^k| \leq k|E^*|$. Nous notons E'^k la solution retournée pour G' par cet algorithme. Nous avons alors $|E'^k| \leq k|E'^*|$. Il y a deux cas possibles.

– Si $|E'^k| \geq x > k|E|$, alors il n'y a pas de solution pour le PROBLÈME DE ROUTAGE pour G. En effet, une telle solution utiliserait au plus les $|E|$ arêtes de G, et une k-approximation aurait utilisé moins de x arêtes.

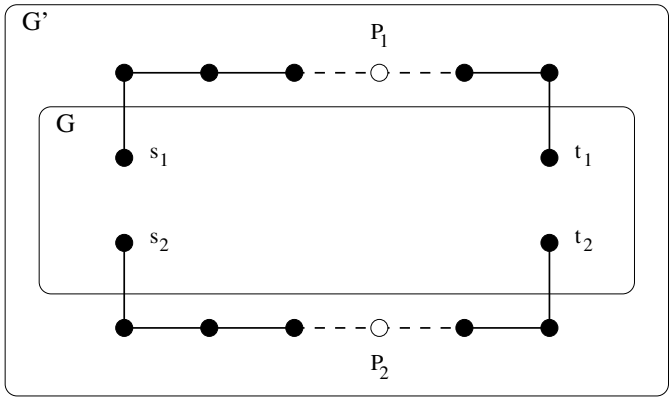


FIG. 3.6 – Graphe G', construit à partir de G, décrit dans la preuve du Théorème 17.

– Si $|E'^k| < x$, alors les deux chemins P_1 et P_2 composés de x arêtes chacun ne sont pas utilisés. Ainsi, $|E'^k| \leq |E|$ et donc il y a une solution pour le PROBLÈME DE ROUTAGE pour G.

En utilisant cette construction, nous aurions alors un algorithme polynomial pour décider s'il y a un routage valide des demandes dans G, en d'autres termes un algorithme polynomial pour le PROBLÈME DE ROUTAGE qui est NP-complet. Une contradiction, à moins que P=NP. □

Théorème 18 *Le* PROBLÈME DE ROUTAGE ARÊTES MINIMUM *n'est pas dans APX même si toutes les arêtes ont une capacité constante c.*

Preuve : Soit $G = (V, E, c)$ le graphe pondéré décrit dans la figure 3.7. Soit $k \geq 1$. Il y a $|\mathcal{D}|$ sommets $s_1, \ldots, s_{|\mathcal{D}|}$ reliés au sommet u par des arêtes de capacité c. Il y a $|\mathcal{D}|$ sommets $t_1, \ldots, t_{|\mathcal{D}|}$ reliés au sommet v par des arêtes de capacité c. Entre u et v, il y a 3 chemins arête-disjoints composés de 2 arêtes chacun et il y a $|\mathcal{D}|$ chemins arête-disjoints composés de $x > k(2|D|+6)$ arêtes chacun. Les $|\mathcal{D}|$ demandes $\mathcal{D}_{s_1t_1}, \mathcal{D}_{s_2t_2}, \ldots, \mathcal{D}_{s_{|\mathcal{D}|}t_{|\mathcal{D}|}}$ sont telles que $\sum_{i=1}^{|\mathcal{D}|} \mathcal{D}_{s_it_i} = 3c$.

Supposons maintenant qu'il existe une constante $k \geq 1$ telle qu'il existe un algorithme polynomial trouvant un sous-ensemble $E^k \subseteq E$, garantissant l'existence d'un routage des demandes utilisant uniquement les arêtes de E^k et assurant $|E^k| \leq k|E^*|$.

– Si $|E^k| \geq x > k(2|D| + 6)$, alors il n'y a pas de partition des demandes en trois sous-ensembles tels que la somme des volumes des demandes de chacun des trois sous-ensembles est c. En effet, si une telle partition existait, l'algorithme garantissant une k-approximation l'aurait trouvée.

– Si $|E^k| < x$, alors les $|D|$ chemins composés de x arêtes chacun ne sont pas utilisés. Ainsi, il y a une partition des demandes en trois sous-ensembles tels que la somme des volumes des demandes de chacun des trois sous-ensembles est c.

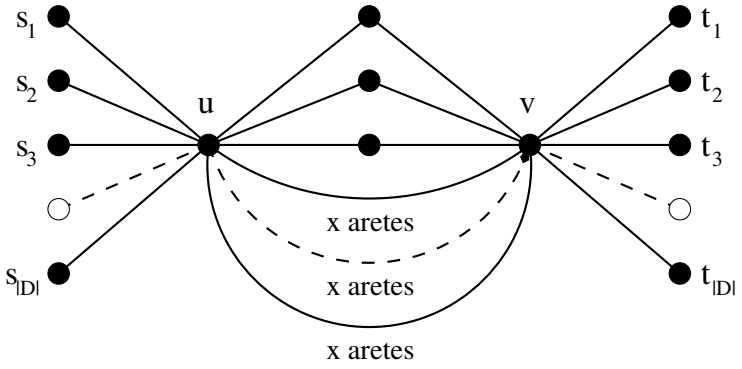


FIG. 3.7 – Graphe G décrit dans la preuve du Théorème 18. Chaque arête $e \in E$ a capacité $c(e) = c$ et $\sum_{i=1}^{|\mathcal{D}|} \mathcal{D}_{s_i t_i} = 3c$.

En utilisant cette construction, nous aurions alors un algorithme polynomial pour résoudre le problème du *Job shop scheduling* qui est NP-complet [GJS76]. Une contradiction, à moins que P=NP. □

3.2.3 Complexité du problème de routage rapport minimum

Nous étudions dans cette section la complexité du problème de déterminer λ_{n-1} dans le cas de demandes de type *one-to-all*.

Théorème 19 *Etant donnés un graphe $G = (V, E)$ et un sommet $r \in V$, le problème de calculer λ_{n-1} dans le cas* one-to-all *est NP-complet.*

Preuve : Soit le graphe $G = (V, E)$ l'union disjointe de k chemins $P_1, P_2, \ldots, P_k$ de tailles $n_1, n_2, \ldots, n_k$, respectivement. Nous notons $u_1, u_2, \ldots, u_k$ une des deux extrémités de $P_1, P_2, \ldots, P_k$, respectivement. Nous posons $n = n_1 + n_2 + \ldots + n_k$. Soit $G' = (V \cup \{r, r_1, r_2, r_3\}, E \cup \{\{r, r_1\}, \{r, r_2\}, \{r, r_3\}\} \cup \{\{r_j, u_i\}, j = 1, 2, 3, i = 1, \ldots, k\})$. Nous considérons des demandes de type *one-to-all* avec r la source. Sans perte de généralité, nous supposons que le volume des demandes est unitaire. G' est représenté dans la figure 3.8.

Nous montrons que $\lambda_{n-1} = \frac{n}{3} + 1$ si, et seulement si, il existe une partition des entiers $n_1, n_2, \ldots, n_k$ en trois sous-ensembles S_1, S_2, S_3 tels que la somme des éléments de chacun des trois sous-ensembles est $\frac{n}{3}$.

Notons tout d'abord que si $\lambda_{n-1} = \frac{n}{3} + 1$, alors les trois arêtes incidentes au sommet r sont dans tout arbre optimal $T^* = (V^*, E^*)$. En effet, si, sans perte de généralité, $\{r, r_1\} \notin E^*$, $\{r, r_2\} \in E^*$ et $\{r, r_3\} \in E^*$, alors la somme des charges des arêtes $\{r, r_2\}$ et $\{r, r_3\}$ est $n + 2$. Ainsi, au moins une des deux arêtes a une charge de $\frac{n}{2} + 1$.

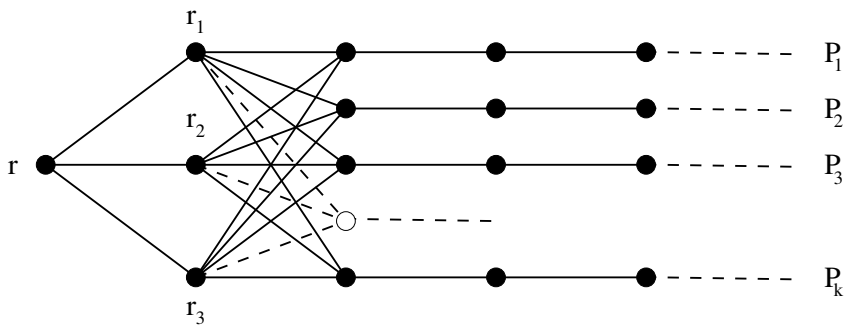


FIG. 3.8 – Graphe G' décrit dans la preuve du Théorème 19.

Supposons donc que $\{r, r_1\}, \{r, r_2\}, \{r, r_3\} \in E^*$. Pour tout $i \in \{1, \ldots, k\}$, une et une seule arête parmi les trois arêtes $\{r_1, u_i\}$, $\{r_2, u_i\}$ et $\{r_3, u_i\}$, est dans T^*. En effet, si deux de ces arêtes sont dans T^* alors il y a un cycle. De plus, toutes les arêtes de P_i sont dans T^*. Pour $j \in \{1, 2, 3\}$, soit l'ensemble $S_j = \{u_i, 1 \leq i \leq k, \{r_j, u_i\} \in E^*\}$. Pour $j \in \{1, 2, 3\}$, la charge de $\{r, r_j\}$ est $1 + \Sigma_{i,u_i \in S_j} n_i$. La somme des charges des arêtes incidentes à r est $3 + n$. Ainsi, le nombre de messages passant par ces trois arêtes est $\frac{n}{3} + 1$ si et seulement si les entiers $n_1, n_2, \ldots, n_k$ peuvent être partitionés en trois sous-ensembles tels que la somme des éléments de chacun est $\frac{n}{3}$. Ce problème (*Job shop scheduling*) étant NP-complet [GJS76], le problème de déterminer λ_{n-1} l'est aussi. □

Nous montrons dans le Théorème 20 que le problème de calculer λ_{n-1} n'est pas dans APX si nous introduisons une contrainte supplémentaire sur les degrés des sommets dans l'arbre solution.

Théorème 20 *Etant donnés un graphe $G = (V, E)$, un degré maximum $d_{max}(u)$ pour chaque sommet $u \in V$ et un sommet $r \in V$, le problème de calculer λ_{n-1} dans le cas* one-to-all *avec la contrainte que $\forall u \in V$, $d_T(u) \leq d_{max}(u)$, n'est pas dans APX ($d_T(u)$ représente le degré de u dans l'arbre solution T).*

Preuve : Soit le graphe $G = (V, E)$ avec $|V| = n$ et v et w deux sommets fixés quelconques. Nous construisons $G' = (V', E')$ à partir de G en attachant à chaque sommet $u \in V$, n^k sommets différents, k une constante. Nous obtenons ainsi G' avec $|V'| = n^{k+1} + n$ sommets. Soient $G'_1, \ldots, G'_{n(q+1)}$, $n(q+1)$ copies de G' contenant respectivement $G_1, \ldots, G_{n(q+1)}$, les $n(q+1)$ copies de G. Nous notons $v_1, \ldots, v_{n(q+1)}$ les copies de v dans $G_1, \ldots, G_{n(q+1)}$, respectivement. De manière analogue, nous notons $w_1, \ldots, w_{n(q+1)}$ les copies de w dans $G_1, \ldots, G_{n(q+1)}$, respectivement. Le graphe $\hat{G} = (\hat{V}, \hat{E})$ est l'union de $G'_1, \ldots, G'_{n(q+1)}$, d'un sommet r et d'un sommet r'. $\forall i \in \{1, \ldots, n(q+1)\}$, il y a une arête entre r et v_i. De plus, il y a une arête entre r' et r. Enfin, $\forall u \in V(G_1) \cup \ldots \cup V(G_{n(q+1)})$, il y a une arête entre r' et u. Rappelons que $\hat{G}$ est construit à partir de copies de G' et donc que chacune de ces copies contient une copie de G. Le nombre de sommets de $\hat{G}$ est $|\hat{V}| = (q+1)(n^{k+2} + n^2) + 2$. La source des

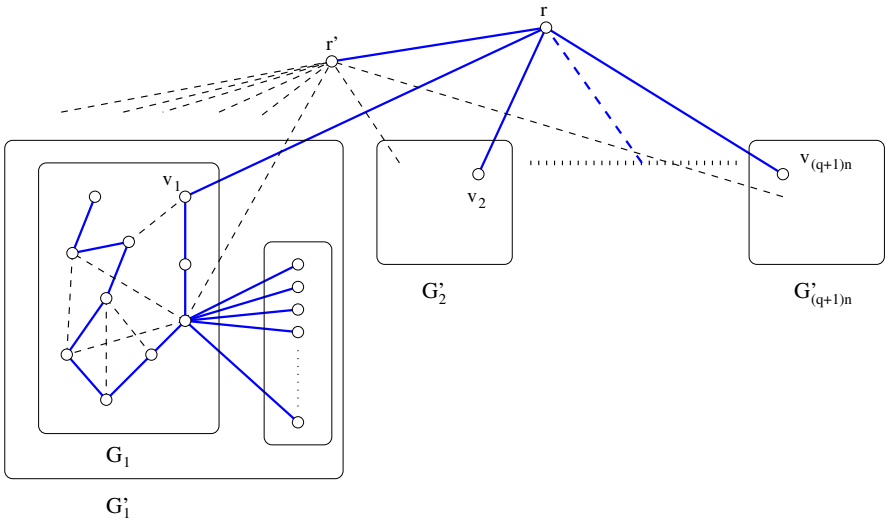


FIG. 3.9 – Graphe $\hat{G}$ de la preuve du Théorème 20 et esquisse d'arbre de charge minimum T^* si le graphe G contient un chemin hamiltonien avec comme extrémités les sommets v et w. T^* est représenté par des traits pleins bleus.

demandes *one-to-all* est le sommet r. Sans perte de généralité, nous supposons que le volume des demandes est unitaire. $\hat{G}$ est représenté dans la figure 3.9. Définissons à présent pour chaque sommet de $|\hat{V}|$, la contrainte sur son degré dans l'arbre. $\forall u \in (V(G_1) \setminus \{w_1\}) \cup \ldots \cup (V(G_{n(q+1)}) \setminus \{w_{n(q+1)}\})$, $d_{max}(u) = n^k + 2$. $\forall u \in \{w_1, \ldots, w_{n(q+1)}\}$, $d_{max}(u) = n^k + 1$. Pour tous les autres sommets u, $d_{max}(u) = \infty$ (aucune contrainte de degré).

Lemme 16 *Si G contient un chemin hamiltonien avec comme extrémités les sommets v et w, alors il existe un arbre couvrant de $\hat{G}$ enraciné en r de charge au plus $n^{k+1} + n$. Sinon, tout arbre couvrant de $\hat{G}$ enraciné en r est de charge au moins $(n^{k+1} + n)(q + 1)$.*

Preuve : (Lemme 16) Supposons que G contient un chemin hamiltonien avec comme extrémités les sommets v et w. Nous construisons alors un arbre $T^* = (V^*, E^*)$ de charge au plus $n^{k+1} + n$ messages. Dans la suite nous considérons $i \in \{1, \ldots, (q+1)n\}$. Nous avons les arêtes $(r, v_i) \in E^*$ et l'arête $(r, r') \in E^*$. Nous construisons maintenant un arbre couvrant de G'_i enraciné en v_i. Un chemin hamiltonien de G_i avec comme extrémités les sommets v_i et w_i est ajouté dans T^*. Rappelons que G_i est une copie de G. De plus, pour tout $u \in V(G'_i) \setminus V(G_i)$, il y a n^k sommets ayant comme unique voisin u. Ainsi, n^k arêtes sont ajoutées dans T^*. Nous pouvons vérifier que tout sommet $u \in V(G_i) \setminus \{w_i\}$ a degré $2 + n^k$ et que w_i a degré $1 + n^k$. Il est facile d'observer que les arêtes de plus grande charge sont (r, v_i). Cette charge est $n^{k+1} + n$, le nombre de sommets de G'_i. T^* est représenté dans la figure 3.9 par des traits pleins bleus.

Supposons que G ne contient pas un chemin hamiltonien avec comme extrémités les sommets v et w. Dans la suite nous considérons $i \in \{1, \ldots, (q+1)n\}$. Soit $T^* = (V^*, E^*)$ un arbre minimisant la charge. Comme il n'existe pas d'arbre couvrant dans G_i respectant les contraintes sur les degrés, il existe au moins un sommet $u \in V(G_i)$ tel que $(r', u) \in E^*$. Si $(r, v_i) \notin E^*$, alors $(r, r') \in E^*$ et la charge de (r, r') est $|\hat{V}| - 1 = (q+1)(n^{k+2} + n^2) + 1$. Supposons maintenant que $(r, v_i) \in E^*$. Nous montrons que $(v_i', r') \notin E^*$ avec $v_i' \in V(G_i)$ un sommet tel qu'il existe un chemin de v_i à v_i' dans T^* et que ce chemin est inclus dans G_i. Sans perte de généralité, supposons que $v_i' = v_i$. Si un telle arête existe dans T^*, nous pouvons mettre en exergue le cycle (r, v_1, r', v_2, r). Une contradiction car T^* est un arbre. La charge de l'arête (r, r') est donc au moins $(n^{k+1} + n)(q+1)$. ⊡

Supposons qu'il existe une constante q telle qu'il existe un algorithme polynomial garantissant que la solution λ' trouvée par ce dernier soit telle que $\lambda' \leq q\lambda_{n-1}$. Le rapport des deux, selon si G contient un chemin hamiltonien avec comme extrémités les sommets v et w ou pas, est $q+1$. Donc, notre algorithme polynomial d'approximation donne un algorithme polynomial qui détermine s'il existe un chemin hamiltonien avec les deux extrémités fixées dans un graphe. Ainsi, nous aurions un algorithme polynomial pour déterminer s'il existe un chemin hamiltonien dans un graphe. En effet, nous pouvons utiliser l'algorithme précédent $\frac{n(n-1)}{2}$ fois, testant toutes les extrémités différentes possibles. Le problème de déterminer s'il existe un chemin hamiltonien dans un graphe est NP-complet [GJ79]. Une contradiction, à moins que P=NP. □

Corollaire 8 *Pour tout entier $k \geq 1$, étant donnés un graphe $G = (V, E)$ à $n = |V|$ sommets, un degré maximum $d_{max}(u) = \Omega(n^{\frac{k}{k+2}})$ pour chaque sommet $u \in V$ et un sommet $r \in V$, le problème de calculer λ_{n-1} dans le cas* one-to-all *avec la contrainte que $\forall u \in V$, $d_T(u) \leq d_{max}(u)$, n'est pas dans APX ($d_T(u)$ représente le degré de u dans l'arbre solution T).*

Nous obtenons ce résultat avec le graphe $\hat{G} = (\hat{V}, \hat{E})$ décrit dans la preuve du Théorème 20. Le nombre de sommets $|\hat{V}|$ est $N = (q+1)(n^{k+2} + n^2) + 2$. Avec q une constante, $N = \theta(n^{k+2})$. La contrainte de degré est $d_{max}(u) = \infty$, $d_{max}(u) = n^k + 2 = \theta(n^k)$ ou $d_{max}(u) = n^k + 1 = \theta(n^k)$, pour tout $u \in V$. Le graphe $\hat{G}$ de la preuve vérifie donc $d_{max}(u) = \Omega(N^{\frac{k}{k+2}})$.

Nous avons prouvé que nous ne pouvions pas approximer λ_{n-1} en ajoutant une contrainte supplémentaire sur le degré dans l'arbre pour un certain sous-ensemble de sommets. Sans cette contrainte, nous ne savons pas si la valeur λ_{n-1} peut être approximée ou non en temps polynomial. Nous conjecturons tout de même que ce problème n'est pas dans APX.

Conjecture 2 *Le problème de calculer λ_{n-1} dans le cas* one-to-all *et* all-to-all *n'est pas dans APX.*

Ces résultats d'inapproximabilité motivent la conception d'heuristiques pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM (section 3.4). Avant cela, nous donnons

dans la section 3.3 des bornes théoriques générales et des bornes pour certaines instances particulières.

3.3 Bornes théoriques

Nous présentons dans cette section des bornes théoriques pour les valeurs des paramètres que nous étudions pour des graphes généraux (section 3.3.1), pour le graphe complet (section 3.3.2) et pour la grille (section 3.3.3).

3.3.1 Graphes généraux

3.3.1.1 Bornes inférieures basées sur la longueur des chemins

La capacité globale du système (somme des capacités des arêtes) doit être nécessairement plus grande que le volume global des demandes (sommes des volumes des demandes). Nous notons $d(s,t)$ la longueur d'un plus court chemin entre une source s et une destination t. Alors

$$\sum_{e \in E} c(e) \geq \sum_{(s,t) \in V \times V, s \neq t} d(s,t) \mathcal{D}_{st}.$$

En particulier, si toutes les arêtes ont la même capacité c et si les demandes sont de type *all-to-all* et de volume κ, alors

$$c|E| \geq \kappa \sum_{(s,t) \in V \times V, s \neq t} d(s,t)$$

et donc

$$\lambda_m \geq \frac{1}{|E|} \sum_{(s,t) \in V \times V, s \neq t} d(s,t).$$

3.3.1.2 Bornes basées sur le flot maximum et la coupe minimum

Pour chaque sous-ensemble $S \subseteq V$, nous devons avoir

$$\sum_{e=\{uv\} \in E; u \in S, v \in \bar{S}} c(e) \geq \sum_{s \in S, t \in \bar{S}} \mathcal{D}_{st} + \sum_{s \in S, t \in \bar{S}} \mathcal{D}_{ts},$$

avec $\bar{S} = V \setminus S$.

En particulier, si toutes les arêtes ont la même capacité c et si les demandes sont de type *all-to-all* et de volume κ, alors

$$c|E_{S\bar{S}}| \geq 2\kappa |S||\bar{S}|$$

et donc

$$\lambda_m \geq \frac{2|S||\bar{S}|}{|E_{S\bar{S}}|}$$

avec $|E_{S\bar{S}}|$ le nombre d'arêtes de la coupe entre S et $\bar{S}$.

Un exemple particulier de coupe est la *coupe bissection minimum* (*minimum bisection cut*), notée $E'_{S\bar{S}}$. La coupe bissection est la coupe minimum qui divise le graphe en deux régions de (presque) mêmes tailles S et $\bar{S}$: $|S| = \lceil \frac{n}{2} \rceil$ et $|\bar{S}| = \lfloor \frac{n}{2} \rfloor$.

En particulier, si toutes les arêtes ont la même capacité c et si les demandes sont de type *all-to-all* et de volume κ, alors nous avons :

$$\lambda_m \geq \frac{2}{|E'_{S\bar{S}}|} \left\lceil \frac{n}{2} \right\rceil \left\lfloor \frac{n}{2} \right\rfloor .$$

Dans la suite, nous supposons que toutes les arêtes ont la même capacité c et que les demandes sont de type *all-to-all* et de volume κ.

3.3.1.3 Borne basée sur le degré maximum du graphe

Afin de présenter une borne générale sur la valeur de λ_{n-1} dans le Lemme 18, nous calculons la valeur de la charge d'un arbre dans le Lemme 17.

Lemme 17 *Soient T un arbre et v la taille de la plus grande branche incidente au centroïde de T. Nous avons $\lambda_m = \lambda_{n-1} \geq 2v(n-v)$.*

Preuve : Supprimer une arête e dans un arbre déconnecte le graphe en deux composantes connexes de tailles $v(e)$ et $n - v(e)$ (avec $v(e) \leq n - v(e)$). La charge de cette arête est $2\kappa v(e)(n - v(e))$.

Considérons le centroïde de l'arbre C et une branche débutant à C. Le centroïde d'un arbre est un sommet ou une arête C qui minimise la plus grande composante connexe induite par la suppression de C du graphe. La charge maximum d'une arête de cette branche est la charge de l'arête e^* incidente à C. En effet, pour tout e de la branche $v(e) < v(e^*)$ et $v(e^*) \leq \lfloor \frac{n+1}{2} \rfloor$, comme toute branche du centroïde est de taille inférieure à $\lfloor \frac{n+1}{2} \rfloor$. Ainsi $2v(e)(n - v(e)) \leq 2v(e^*)(n - v(e^*))$ car la fonction $x(n-x)$ est croissante pour $0 \leq x \leq \frac{n}{2}$. Pour la même raison, la charge de l'arbre est la charge de l'arête du centroïde (si le centroïde est une arête) ou incidente au centroïde de la plus grande branche. □

Lemme 18 *Pour un graphe G de degré maximum $\Delta(G)$, nous avons :*

$$\lambda_{n-1} \geq 2 \left\lceil \frac{n-1}{\Delta(G)} \right\rceil \left(n - \left\lceil \frac{n-1}{\Delta(G)} \right\rceil \right) .$$

Preuve : La charge d'un arbre est la charge de l'arête centroïde ou incidente au centroïde de la plus grande branche. La charge est alors minimum pour un arbre avec la plus grande branche minimum. Pour un graphe G de degré maximum $\Delta(G)$, la plus grande branche d'un arbre couvrant est de taille au moins $\lceil \frac{n-1}{\Delta(G)} \rceil$. Ainsi, $\lambda_{n-1} \geq 2\kappa \lceil \frac{n-1}{\Delta(G)} \rceil (n - \lceil \frac{n-1}{\Delta(G)} \rceil)$. □

Nous observons que la charge d'un arbre (couvrant) dépend sensiblement du degré maximum du graphe. Par exemple pour le graphe complet, avec $\Delta = n - 1$,

l'arbre de charge minimum est une étoile et la charge vaut $2\kappa(n-1)$. En comparaison, si l'arbre est un chemin (degré maximum $\Delta = 2$), sa charge vaut alors $\kappa\lfloor\frac{n^2}{2}\rfloor$. Les réseaux avec des sommets de grands degrés semblent atteindre une configuration minimale pour de petites valeurs des capacités (voir section 3.4.3).

3.3.2 Graphe complet

Nous considérons dans cette section le graphe complet K_n composé de n sommets et de $n(n-1)/2$ arêtes. Cette topologie représente un cas extrême de notre étude et correspond au *problème de design* dans lequel il s'agit de concevoir le meilleur réseau satisfaisant la charge lorsque toutes les arêtes possibles peuvent être utilisées. Nous supposons que les demandes sont *all-to-all* de volume constant κ et que chaque arête a une capacité constante c.

Le routage est possible dans K_n dès que $\lambda \geq 2$. De plus, si $\lambda \geq 2(n-1)$, alors le routage est possible avec une étoile composée de $n-1$ arêtes. Cela correspond à une fraction $2/n$ du nombre total d'arêtes de K_n. En d'autres termes, nous avons $\lambda_m = 2$ et $\lambda_{n-1} = 2(n-1)$. Une question importante est alors d'étudier le nombre d'arêtes pouvant être économisées pour des valeurs de λ comprises entre 2 et $2(n-1)$.

Lemme 19 *Etant donnés un graphe complet pondéré $K_n = (V, E, c)$, $c(e) = c$ pour tout $e \in E$, et un ensemble de demandes de type* all-to-all *chacune de volume κ, toute solution optimale E^* pour le* PROBLÈME DE ROUTAGE ARÊTES MINIMUM *est telle que* $|E^*| \geq \max\left(\frac{\kappa n(n-1)}{c}, n-1\right)$.

Preuve : Dans la section 3.3.1, nous avons vu que

$$|E^*|c \geq \kappa \sum_{i,j \in V \times V} d(i,j),$$

avec $d(i,j)$ la distance entre i et j. Notons que $\sum_{i,j \in V \times V} d(i,j)$ est donné par une formule close comme deux fois l'*indice de Wiener* (*Wiener index*) [Wie47, Wei].

Dans un graphe complet, tous les plus courts chemins ont taille 1. Si nous supprimons l'arête ij, deux chemins (de i à j et de j à i) sont maintenant de taille 2. Nous en déduisons que pour un sous-graphe optimal avec $|E^*|$ arêtes, au plus $2|E^*|$ chemins ont taille 1 et $n(n-1) - 2|E^*|$ chemins ont taille au moins 2. Alors

$$|E^*|c \geq \kappa \sum_{ij \in V \times V} d(i,j) \geq 2\kappa|E^*| + \kappa(n(n-1) - 2|E^*|).$$

La borne donne

$$|E^*| \geq \frac{\kappa n(n-1)}{c}.$$

□

Dans la figure 3.10, nous présentons la valeur $\frac{m-|E^*|}{m}$ donnée par le programme linéaire entier [GMMO10b] pour un graphe complet composé de 5 sommets (avec

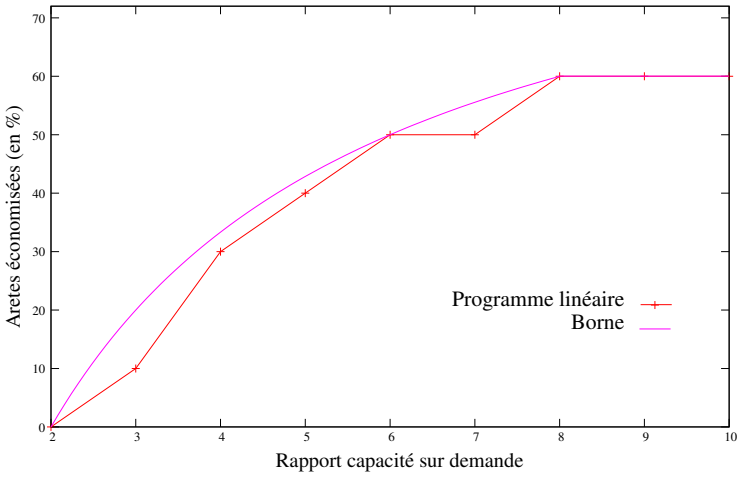


FIG. 3.10 – Pourcentages d'arêtes économisées pour le graphe complet à 5 sommets en fonction de λ. Borne du Lemme 19 et valeur du programme linéaire entier [GMMO10b].

CPLEX 10). La figure montre que la borne inférieure donnée dans le Lemme 19 est proche de la solution optimale. Le rapport λ varie de 2 à 8 comme stipulé précédemment. Par exemple, pour $\lambda = 4$, nous avons 30% d'arêtes économisées (7 arêtes activées au lieu de 10).

3.3.3 Grille

Nous considérons dans cette section le cas particulier d'une topologie en grille qui représente un exemple de réseau structuré classiquement étudié, par exemple dans les réseaux sans-fil [AWW05]. Nous supposons que les demandes sont *all-to-all*. Dans la section 3.3.3.1, nous étudions tout d'abord les valeurs λ_{n-1} et λ_m correspondant aux deux configurations limites : l'arbre couvrant de charge minimum (Lemme 20) et le sous-graphe (sans contrainte) de charge minimum (Lemme 21), respectivement. Nous nous intéressons dans la section 3.3.3.2 aux différentes configurations intermédiaires. Nous proposons une construction d'un sous-graphe de la grille lorsque le nombre d'arêtes de ce sous-graphe est fixé (Lemme 22). Cela donne une borne supérieure sur le rapport λ sous cette contrainte. Nous prouvons également une borne inférieure pour la charge d'un sous-graphe de la grille avec contrainte sur le nombre d'arêtes (Lemme 23). Dans la section 3.4.2, nous comparons ces constructions aux solutions données par le programme linéaire entier et les heuristiques que nous proposons dans la section 3.4.1.

3.3.3.1 Configurations limites

La valeur du rapport λ_{n-1} correspondant à la charge minimum d'un arbre couvrant la grille, est donnée dans le Lemme 20.

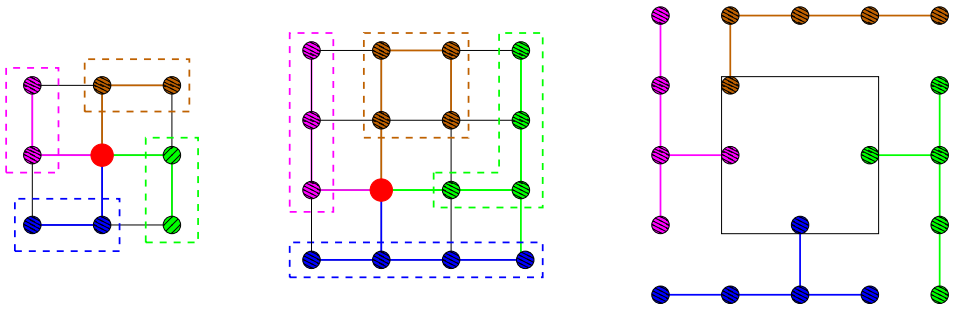


FIG. 3.11 – Partition de la grille en 4 sous-ensembles connexes de tailles presque identiques. Gauche : grille 3×3. Milieu : grille 4×4. Droite : induction de la preuve du Lemme 20.

Lemme 20 *Pour une grille $a \times a$ ($a \geq 3$), nous avons :*

$$\lambda_{n-1} = 2 \left\lceil \frac{a^2-1}{4} \right\rceil \left(a^2 - \left\lceil \frac{a^2-1}{4} \right\rceil \right) \approx \frac{3a^4}{8}.$$

L'arbre permettant d'obtenir cette valeur est un arbre avec un centroïde de degré 4 et 4 branches de tailles (presque) identiques.

Preuve : Dans toute grille $a \times a$, il est possible de partitioner les sommets en un sommet central C et 4 sous-ensembles connexes de tailles $\lceil \frac{a^2-1}{4} \rceil$ et $\lfloor \frac{a^2-1}{4} \rfloor$. La preuve se fait par induction. L'hypothèse d'induction H_a est : la partition $P = \{P_1, P_2, P_3, P_4\}$ existe pour une grille $a \times a$ et la partie i de la partition est adjacente au côté i de la grille. Cela est vrai pour une grille 3×3 et une grille 4×4 (voir figure 3.11). Nous prouvons que H_a implique H_{a+2} en connectant le côté i du carré à un sommet de la partie P_i qui est sur le côté i.

Il existe un arbre couvrant de centroïde C avec quatre branches et de plus grande branche de taille $\lceil \frac{a^2-1}{4} \rceil$. Sa charge est alors $2\kappa \left\lceil \frac{a^2-1}{4} \right\rceil \left(a^2 - \left\lceil \frac{a^2-1}{4} \right\rceil \right)$ et est maximum, comme décrit dans le Lemme 18. □

Nous prouvons dans le Lemme 21 une borne inférieure pour λ_m.

Lemme 21 *Dans une grille $a \times a$, nous avons :*

$$\lambda_m \geq \begin{cases} \frac{a^3}{2} & \text{si } a \text{ est pair,} \\ \frac{a^3-a}{2} & \text{si } a \text{ est impair.} \end{cases}$$

Preuve : Nous divisons la grille en deux sous-ensembles S et $\bar{S}$ de tailles (presque) identiques.

– Si a est pair, nous avons $|S| = |\bar{S}| = \frac{a^2}{2}$. Nous avons a arêtes dans la coupe. Ainsi, nous obtenons $ac \geq 2\kappa \frac{a^2}{2} \frac{a^2}{2}$ qui donne $\lambda_m \geq \frac{a^3}{2}$.

– Si a est impair, nous avons $|S| = \frac{(a-1)a}{2}$ et $|\bar{S}| = \frac{(a+1)a}{2}$. Nous avons a arêtes dans la coupe. Ainsi, $ac \geq 2\kappa \frac{(a-1)a}{2} \frac{(a+1)a}{2}$ qui donne $\lambda_m \geq \frac{a^3-a}{2}$.

□

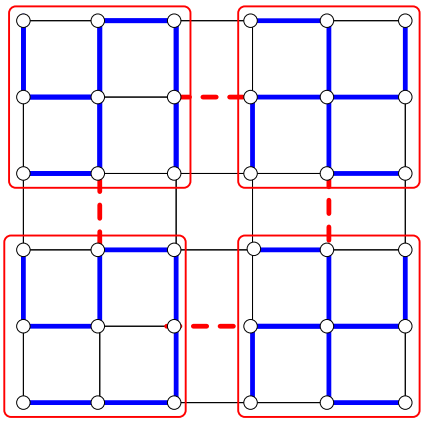


FIG. 3.12 – Construction d'un sous-graphe avec $n+p-\sqrt{p}$ arêtes. **1)** diviser la grille en p régions de tailles presque identiques ($p = 4$ dans notre exemple). **2)** relier les régions en grille (arêtes rouges en pointillés). **3)** Connecter les sommets d'une région en arbre (arêtes bleues larges).

3.3.3.2 Configurations intermédiaires

Nous avons prouvé dans la section précédente les valeurs des deux charges extrêmes pour la grille. La charge λ varie donc de $\frac{n^{3/2}}{2}$ à $\frac{3n^2}{8}$. Nous étudions l'évolution de cette charge entre ces deux configurations limites. Nous donnons une borne supérieure sur la charge d'un sous-graphe si le nombre d'arêtes est contraint et fixé (Lemme 22). Nous prouvons ensuite une borne inférieure sur la charge lorsque le nombre d'arêtes pouvant être utilisées dans le routage est fixé (Lemme 23). La preuve est basée sur les plus courts chemins de la grille.

Dans le Lemme 22, nous présentons une construction de sous-graphe composé de $n+p-2\sqrt{p}$ arêtes pour p un carré, c'est-à-dire si $\exists q \in \mathbb{N}, p = q^2$. La charge de ce sous-graphe est de l'ordre de $n^2/\sqrt{p}$. Cette construction donne un borne supérieure sur la charge d'un sous-graphe de la grille ayant un nombre contraint d'arêtes. Pour les autres valeurs de p ($-1 \leq p \leq n$), cette formule est prise comme une approximation de la charge du meilleur sous-graphe. Dans la section 3.4.2, nous analysons la qualité de cette approximation pour deux exemples : la grille 4×4 et la grille 10×10.

Lemme 22 *Soient G une grille composée de n sommets et p un carré entre 1 et n. Nous avons $\lambda_{n+p} \leq \frac{1}{2}\frac{n^2}{\sqrt{p}} + \frac{3}{8}\frac{n^2}{p^2}$.*

Preuve : Nous construisons un graphe de la charge désirée. Nous divisons la grille en p régions (figure 3.12). Les régions sont connectées en *grille carrée*. Les sommets d'une région sont liés via un *arbre*. Le graphe a $n+p-2\sqrt{p}$ arêtes ($n-1-(p-1)$ arêtes à l'intérieur des différentes régions et $2p-2\sqrt{p}$ arêtes de la grille). Remarquons que lorsque $p = n$, la construction est simplement la grille entière. De plus lorsque $p = 1$, la construction est un arbre couvrant.

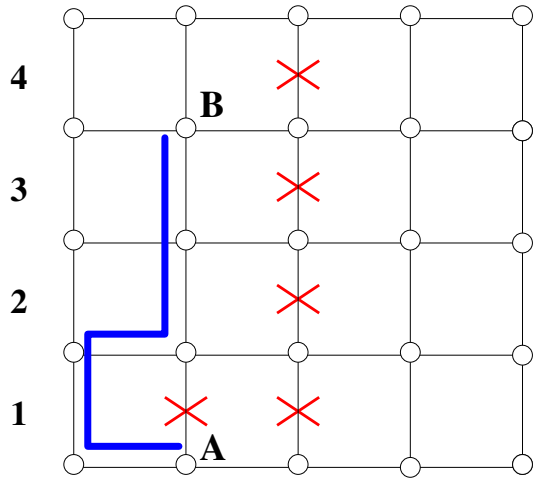


FIG. 3.13 – Preuve du Lemme 23. Une grille avec 5 arêtes supprimées : les arêtes sont supprimées dans une colonne (ligne) partant de la position 1. La longueur du plus court chemin entre A et B augmente de 2 (chemin bleu large).

Charge des arêtes de la grille carré, L_{grid}. La charge d'une grille de p sommets est $\frac{1}{2}p^{3/2}$. Chaque région correspond à $\lceil \frac{n}{p} \rceil$ ou $\lfloor \frac{n}{p} \rfloor$ sommets. Ainsi, entre deux sommets, il y a (au plus) $\lceil \frac{n}{p} \rceil \times \lceil \frac{n}{p} \rceil$ routes au lieu de 1. Nous pouvons router avec une capcité de $\frac{1}{2}\frac{n^2}{p^2}p^{3/2}$ en choisissant le routage de la grille pour les $\approx \frac{n^2}{p^2}$ demandes.
Charge des arêtes à l'intérieur des régions, L_{region}. Il y a au plus $4L_{grid}$ demandes entrantes/sortantes d'un sommet. Ces demandes doivent être routées via la région. L'arbre couvrant d'une région peut être construit tel que les (au plus) 4 arêtes de la grille sont connectées par 4 branches à un sommet de la région de degré 4. Ainsi, il est possible de router cette demande avec une capacité de L_{grid}.

De plus, les demandes doivent être routées en deux sommets de la région. Cela correspond au routage *all-to-all* dans un arbre de $\lceil \frac{n}{p} \rceil$. Une capacité additionnelle de $\approx \frac{3}{8}n^2$ est nécessaire.

En conclusion, nous obtenons $L_{region} \leq \frac{1}{2}\frac{n^2}{\sqrt{p}} + \frac{3}{8}\frac{n^2}{p^2}$ □

Nous prouvons dans le Lemme 23 une borne inférieure sur la charge d'un sous-graphe de la grille lorsque le nombre d'arêtes est fixé.

Lemme 23 *Soit G une grille $a \times a$. Nous avons*

$$\lambda_k \geq \frac{1}{k}\left(\frac{2a^3(a^2-1)}{3} + \left\lfloor \frac{|E|-k}{a} \right\rfloor 2a(a-1) + 4\sum_{i=1}^{|E|-k-\left\lfloor \frac{|E|-k}{a} \right\rfloor (a-1)} (a-i)\right)$$

Preuve : La somme des tailles d'un plus court chemin d'une grille est donnée par l'*indice de Wiener* (*Wiener index*) [Wie47, Wei] d'une grille. Nous multiplions cette valeur par deux car dans notre problème, nous avons des demandes *all-to-all* : nous avons à la fois le chemin de u vers v et le chemin de v vers u. Nous obtenons $\frac{2a^3(a^2-1)}{3}$.

Algorithme 4 : `Heuristique de l'arête la moins chargée`

Précondition : un graphe pondéré $G = (V, E, c)$ dans lequel chaque arête $e \in E$ a une capacité initiale $c(e)$ et une capacité résiduelle r_e, et un ensemble de demandes $\mathcal{D}$ dans lequel chaque demande a un volume de trafic $\mathcal{D}_{st}$.

$\forall e \in E, r_e = c(e)$
Calculer un routage valide des demandes avec l'Algorithme 5
tant que des arêtes peuvent être supprimées **faire**
  supprimer l'arête e' qui n'a pas déjà été supprimée ayant la plus petite valeur $\frac{c(e')}{r(e')}$.
  Calculer un routage valide des demandes avec l'Algorithme 5
  Mettre à jour les capacités résiduelles $r_e, \forall e \in E$
  S'il n'y a pas de routage valide, alors réinsérer e' dans G
retourner le sous-graphe G.

Lorsque nous supprimons une arête de la grille, tous les plus courts chemins qui utilisent cette arête voient leurs tailles augmenter de 2 unités (figure 3.13). Pour une arête à la position i, il y a au moins $2i(a-i)$ tels chemins sur la colonne (ou ligne) de l'arête. Cette valeur est minimum lorsque $i = 1$ et lorsque $i = a-1$. Pour obtenir une borne inférieure sur la charge, nous devons choisir une séquence de suppressions d'arêtes qui minimise l'augmentation de la somme des plus courts chemins. Il est optimal de supprimer les arêtes d'une colonne (ligne) avec des arêtes manquantes. La deuxième suppression d'arêtes augmente la taille d'au moins $2(a-2)$ chemins et plus généralement la i^e suppression augmente la taille de $2(a-i)$ chemins. Au total, au moins $a(a-1)$ chemins sont augmentés lorsqu'une colonne (ligne) a été supprimée.

Dans un sous-graphe composé de k arêtes, après la suppression de $|E|-k$ arêtes, $\left\lfloor \frac{|E|-k}{a} \right\rfloor$ telles colonnes (lignes) peuvent être supprimées. Nous obtenons le terme $2\left\lfloor \frac{|E|-k}{a} \right\rfloor a(a-1)$. Les arêtes restantes sont supprimées de la même colonne (ligne) donnant le terme $2\sum_{i=1}^{|E|-k-\left\lfloor \frac{|E|-k}{a} \right\rfloor (a-1)}(2a-2i)$. □

3.4 Heuristiques et simulations

3.4.1 Heuristiques

Comme nous l'avons vu dans la section 3.2, le PROBLÈME DE ROUTAGE ARÊTES MINIMUM est un problème très difficile à résoudre exactement et même à approximer à un facteur constant près. Il est alors nécessaire de proposer des heuristiques dans le but d'obtenir des résultats pour des topologies de réseaux réelles. Nous proposons deux heuristiques : `Heuristique de l'arête la moins chargée` et `Heuristique aléatoire`.

L'Algorithme 4 décrit en détails l'`Heuristique de l'arête la moins`

Algorithme 5 : HEURISTIQUE ROUTAGE VALIDE

Précondition : un graphe pondéré $G = (V, E, c)$ dans lequel chaque arête $e \in E$ a une capacité initiale $c(e)$ et une capacité résiduelle r_e, et un ensemble de demandes $\mathcal{D}$ dans lequel chaque demande a un volume de trafic $\mathcal{D}_{st}$.

Trier les demandes aléatoirement
tant que $\mathcal{D}_{st}$ est une demande de $\mathcal{D}$ sans route assignée **faire**
Calculer un plus court chemin SP_{st} avec la métrique $\frac{c(e)}{r_e}$ sur les arêtes
Assigner le routage SP_{st} à la demande $\mathcal{D}_{s,t}$
$\forall e \in SP_{st}, r_e = c(e) - \mathcal{D}_{st}$
retourner le routage (s'il existe) assigné aux demandes de $\mathcal{D}$.

`chargée` pour le PROBLÈME DE ROUTAGE ARÊTES MINIMUM. A partir du réseau initial, nous calculons un routage valide en utilisant l'Algorithme 5 et essayons de supprimer les arêtes les moins chargées (les moins utilisées) dans ce routage. Nous pensons que supprimer des arêtes qui ne sont pas dans beaucoup de plus courts chemins va engendrer le reroutage d'un petit nombre de demandes dans le réseau. Concernant le routage, les demandes sont considérées une par une et dans un ordre aléatoire. Nous calculons un plus court chemin avec la métrique $\frac{c(e)}{r_e}$ pour les arêtes, avec r_e la capacité résiduelle de l'arête e. Ensuite la capacité résiduelle est mise à jour pour chacune des arêtes et la prochaine demande est alors considérée. Notons que la recherche d'un routage valide peut également se faire avec un programme linéaire entier pour le PROBLÈME DE ROUTAGE si les topologies sont suffisamment petites. A chaque fois qu'une arête est supprimée du graphe, un routage valide est calculé. Si un tel routage n'existe pas ou n'est pas trouvé par l'algorithme ou si la solution a un coût supérieur à celle trouvée à l'étape précédente, alors l'arête supprimée est réintégrée au graphe et nous essayons avec une autre arête qui n'a pas encore été considérée. L'algorithme termine lorsque toutes les demandes ont été considérées.

La seconde heuristique, `Heuristique aléatoire`, est analogue à l'`Heuristique de l'arête la moins chargée`. La seule différence est que dans l'`Heuristique aléatoire` les arêtes sont supprimées uniformément au hasard (et donc pas nécessairement celles les moins chargées). Le routage est calculé de la même manière.

3.4.2 Résultats de simulations pour la grille

Nous présentons des résultats pour la grille 4×4 (programme linéaire entier, construction du Lemme 22, borne inférieure du Lemme 23 et heuristiques) et pour la grille 10×10 (sans le programme linéaire entier).

Borne basée sur la coupe minimum. Dans la figure 3.14, nous présentons des résultats pour la grile 4×4. Nous avons tracé la borne du Lemme 19 (− magenta). Nous observons que, contrairement aux graphes complets, cette borne n'est pas atteinte. En effet, les arêtes de la grille ne sont pas toutes équivalentes : les arêtes

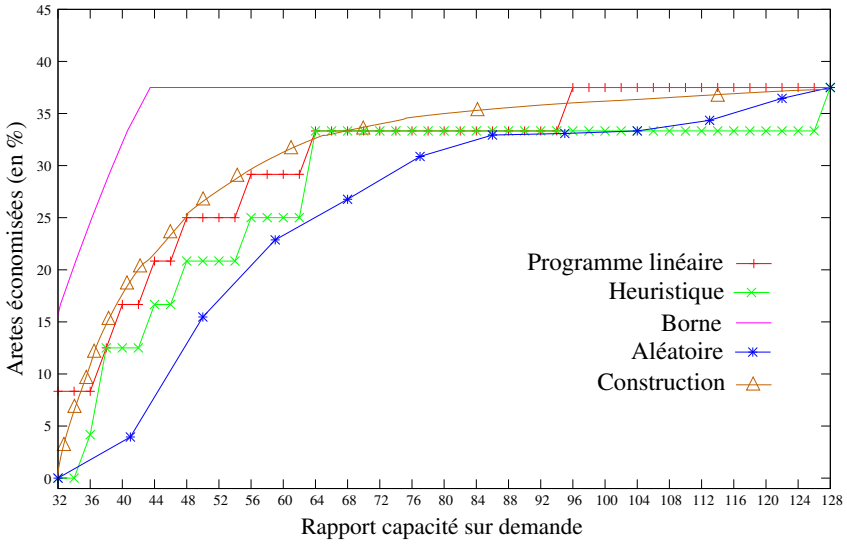


FIG. 3.14 – Pourcentages d'arêtes économisées pour une grille 4×4 en fonction de λ.

du milieu de la grille ont une charge plus importante que celles du bord avec un routage par plus court chemin.

La charge peut être calculée à partir de la *coupe minimum* du graphe. La coupe composée des a arêtes horizontales du milieu de la grille donne $\lambda_m \geq \frac{a^3}{2}$. Ce phénomène de coupe explique également que dès lors que le routage est possible (pour $\lambda = 32$), un certain nombre d'arêtes peuvent déjà être économisées (environ 8%) : les arêtes du milieu sont totalement chargées mais pas celles des bords. Ce même phénomène de coupe est observable pour des topologies réelles étudiées dans la section 3.4.3.

Ordre donné par la construction. Cette ordre, donné dans le Lemme 22, est décrit dans la figure 3.14 et dans la figure 3.15. Nous voyons que les valeurs sont très proches des valeurs optimales données par le programme linéaire entier et largement meilleures que la borne utilisant la coupe minimum. Pour la grille 10×10, nous ne pouvons pas utiliser le programme linéaire entier mais nous observons que l'`Heuristique de l'arête la moins chargée` se comporte très bien comparée à cet ordre.

Nombre d'arêtes qui peuvent être économisées. Pour des valeurs de λ supérieures à 32, un grand nombre d'arêtes peuvent être économisées au début : pour une grille 4×4, 25% pour $\lambda = 48$ et 33% pour $\lambda = 64$. L'économie devient moins importante pour des valeurs de λ plus grandes : seulement 4% points de pourcentage de différence pour des valeurs de λ comprises entre 64 et 96, et l'arbre est atteint pour $\lambda = 96$. Pour cette valeur de λ, $a^2 - 1 = 15$ arêtes sont utilisées, alors que la grille entière a 24 arêtes, économisant 37.5% d'arêtes.

Comportements des heuristiques. Nous observons que l'`Heuristique de`

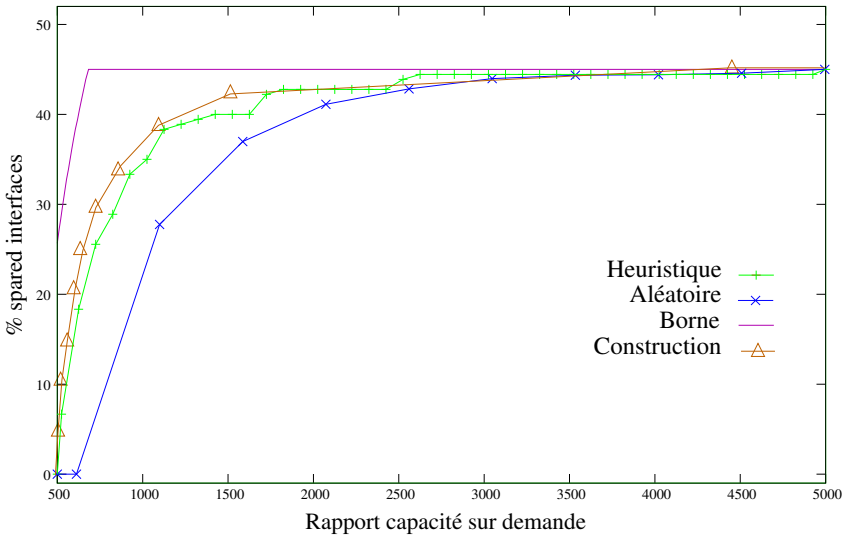


FIG. 3.15 – Pourcentages d'arêtes économisées pour une grille 10×10 en fonction de λ.

`l'arête la moins chargée` ($\times$ vert) donne des solutions très proches des valeurs optimales données par le programme linéaire entier ($+$ rouge). De plus, l'`Heuristique de l'arête la moins chargée` est aussi significativement meilleure que l'`Heuristique aléatoire` (figure 3.14 et figure 3.15). Par exemple, pour la grille 10×10 et pour $\lambda = 1000$, l'`Heuristique de l'arête la moins chargée` économise 39% alors que l'`Heuristique aléatoire` seulement 28%.

3.4.3 Résultats de simulations pour des topologies réelles

Nous supposons que les capacités de toutes les arêtes sont c et que les demandes sont de type *all-to-all* avec un volume κ. En utilisant nos heuristiques, nous étudions combien d'arêtes peuvent être économisées pour dix réseaux classiques extraits de la librairie SNDLib[2] selon la valeur de $\lambda = \frac{c}{\kappa}$.

Dans le tableau 3.1, nous donnons le pourcentage d'arêtes économisées pour différentes valeurs de λ. Nous remarquons que pour certains graphes (topologies), des arêtes peuvent être économisées même pour $\lambda = \lambda_m$ (12% pour *Norway* et *Nobel EU*). Lorsque $\lambda = 2\lambda_m$ (correspondant par exemple à un trafic de nuit deux fois moins important), nous observons qu'environ un tiers des arêtes peuvent être économisées (et même 53% pour le réseau *Pioro40*). En augmentant encore la valeur de λ, nous remarquons que le gain marginal devient faible. Les deux dernières colonnes mettent en exergue λ_{n-1} et le pourcentage d'arêtes économisées. Par exemple pour le réseau *Atlanta* (figure 3.16(a)), nous obtenons la valeur $\lambda_{n-1} = 2.66\lambda_m$ en prenant l'arbre représenté dans la figure 3.16(b).

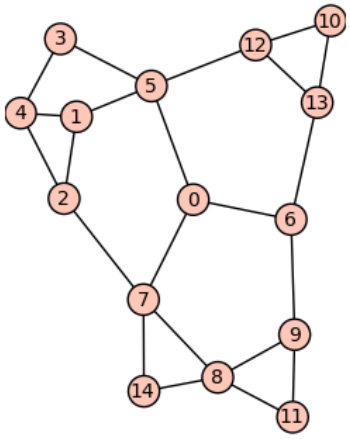
[2] `http://sndlib.zib.de`

Réseau			Gain pour $\lambda =$				Arbre	
	n	m	λ_m	$2\lambda_m$	$3\lambda_m$	$4\lambda_m$	$\frac{\lambda_{n-1}}{\lambda_m}$	Gain
Atlanta	15	22	0%	32%	36%	36%	2.66	36%
New York	16	49	2.0%	59%	63%	67%	5.2	69%
Nobel Germany	17	26	0%	35%	39%	39%	2.75	39%
France	25	45	0%	42%	44%	47%	3.13	47%
Norway	27	51	12%	43%	47%	47%	4.71	49%
Nobel EU	28	41	12%	32%	34%	34%	2.76	34%
Cost266	37	57	3.5%	32%	35%	37%	3.68	37%
Giul39	39	86	0%	45%	50%	52%	8.25	56%
Pioro40	40	89	0%	53%	54%	55%	5.12	56%
Zib54	54	80	0%	30%	33%	33%	4.71	34%

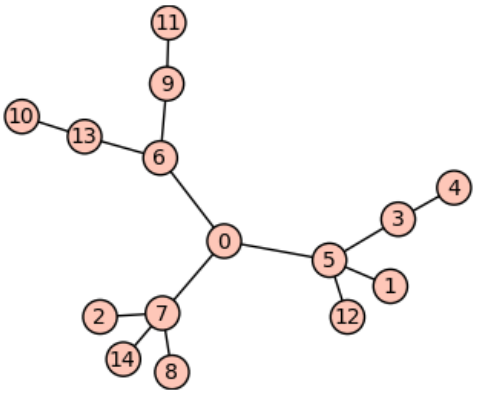
TAB. 3.1 – Pourcentages d'arêtes économisées pour $\lambda = \lambda_m$, $\lambda = 2\lambda_m$, $\lambda = 3\lambda_m$ et $\lambda = 4\lambda_m$. Valeurs $\frac{\lambda_{n-1}}{\lambda_m}$ et les pourcentage d'arêtes économisées associés.

Réseau	Simulations		Valeurs obtenues avec les bornes			
			λ_m			λ_{n-1}
	λ_m	λ_{n-1}	Coupe	Borne	$\Delta(G)$	Borne
Atlanta	38	101	3	38	4	88
New York	15	78	12	11	11	56
Nobel Germany	44	121	4	36	5	104
France	67	210	7	45	10	132
Norway	75	354	6	61	6	220
Nobel EU	131	362	3	131	5	264
Cost266	175	644	4	171	4	540
Giul39	85	702	11	70	8	340
Pioro40	153	784	7	115	5	512
Zib54	294	1385	6	243	10	576

TAB. 3.2 – Evaluation de λ_m et λ_{tree}. Ces deux valeurs sont comparées avec les bornes théoriques de la section 3.3.1.



(a) Graphe représentant le réseau *Atlanta*.



(b) Arbre de charge minimum pour le graphe représentant le réseau *Atlanta* : $\lambda_{n-1} = 101$.

FIG. 3.16 – Graphe représentant le réseau *Atlanta* et arbre de charge minimum associé.

Dans le tableau 3.2, nous représentons la valeur λ_m. Comme décrit dans la section 3.3.1, λ_m dépend fortement de la coupe minimum du graphe. Nous calculons cette coupe minimum en utilisant un programme linéaire, puis nous en déduisons une borne inférieure pour λ_m. Nous remarquons que cette dernière est très proche (parfois identique) de la valeur obtenue par les simulations (même si cette dernière n'est pas toujours optimale). De façon analogue, nous obtenons par simulations la valeur λ_{n-1}. Nous présentons également le degré maximum $\Delta(G)$ de chaque graphe G ainsi que la borne induite par celle-ci sur λ_{n-1}. Nous remarquons des différences relativement importantes par rapport aux valeurs obtenues par simulations.

3.4.3.1 Impact des solutions proposées

Impact sur la longueur des routes. Nous remarquons que les routages obtenus ont un impact limité sur l'augmentation de la longueur des routes. Pour *Zib54*, l'accroissement est de 11% pour le cas extrême où le routage se fait avec un arbre (34% d'économie). En général, l'augmentation pour ce cas extrême varie de 11% à 50% avec une moyenne de 27%. Concernant le réseau le plus impacté *Giul39* avec un accroissement de 50% de la longueur des routes, l'économie est de 56%. Nous observons qu'une économie d'arêtes de 45% est atteinte pour $\lambda = 2\lambda_m$ avec une augmentation de la longueur des routes de seulement 18%.

Impact sur la tolérance aux pannes. Nous mesurons l'impact sur la tolérance aux pannes en calculant *le nombre moyen de chemins disjoints* entre deux sommets. Ce nombre varie de 1.82 à 4.90 pour les réseaux complets. En routant les demandes via un arbre, ce nombre est 1 par définition (une seule route existe entre n'importe

quelle paire de sommets). Cette valeur est presque atteinte avec $\lambda = 3\lambda_m$. Nous observons enfin que cette diminution est rapide car pour les dix réseaux, cette valeur est inférieure à 1.41 pour $\lambda = 2\lambda_m$.

3.5 Conclusion et perspectives

D'un point de vue théorique, nous tentons actuellement de savoir si le problème de déterminer la valeur λ_{n-1} est dans APX ou non (Conjecture 2). Nous étudions également les valeurs des différents paramètres pour d'autres topologies particulières.

Nous travaillons également sur une modélisation du problème permettant de mieux en prendre compte la question de la tolérance aux pannes et de la longueur des routes [GMMO10a].

D'un point de vue pratique, nous sommes en train d'expérimenter nos solutions sur une plateforme d'expérimentation composée d'une vingtaine de machines. Nous étudierons également ce problème en utilisant une architecture de réseau un peu plus complexe.

Chapitre 4

Ordonnancement des liens dans les réseaux sans-fil

Sommaire

4.1 Introduction

Ce travail a été réalisé avec Jean-Claude Bermond, Vishal Misra et Philippe Nain [BMMN10, BMMN08, BMMN09].

4.1.1 Motivations

L'ordonnancement des transmissions est un problème central dans les réseaux de télécommunication. Un grand nombre de travaux a été réalisé pour les réseaux filaires ainsi que les réseaux sans-fil (réseau radio, réseau ad hoc, réseau de capteurs). Dans les réseaux sans-fil, une des difficultés majeures réside dans les problèmes d'interférence. En effet, durant une étape (ou *time slot*), seules les transmissions n'interférant

pas entre elles peuvent être ordonnancées. Dans ce chapitre, nous considérons tous les modèles d'interférence binaire. Cependant pour les calculs, pour les exemples et pour les simulations, nous utilisons le modèle d'interférence à distance d, dans lequel deux transmissions interfèrent si et seulement si la distance entre les deux est plus petite que d.

Comme les processus d'arrivées ne sont pas nécessairement connus a priori par les concepteurs du réseau, les algorithmes doivent déterminer l'ensemble des liens actifs en fonction des arrivées passées et présentes. Cependant seuls des algorithmes distribués avec une connaissance locale limitée peuvent être utilisés en pratique. En effet, les algorithmes centralisés sont basés sur une connaissance totale du réseau à chacune des étapes, une information très difficile à obtenir en raison des interférences. En effet, le nombre de messages de contrôle de tels algorithmes centralisés n'est pas constant en la taille du réseau. Ce problème est en fait au moins aussi difficile que le problème d'ordonnancement des liens. De plus, dans le but d'être efficace en énergie (énergie limitée des capteurs par exemple), les décisions doivent se prendre localement, sans échange de messages avec une station de base. Enfin, ces algorithmes doivent pouvoir supporter différents modèles d'interférence binaire.

4.1.2 Modélisation

4.1.2.1 Le réseau est modélisé par un graphe

Nous représentons le réseau par un digraphe (des transmissions) orienté symétrique $G = (V, A)$ avec V représentant l'ensemble des nœuds du réseau et il y a un arc $(u, v) \in A$ (et donc un arc $(v, u) \in A$) entre deux sommets $u \in V$ et $v \in V$ si et seulement si ces deux derniers peuvent communiquer. En effet, nous supposons que la relation est symétrique : si u peut transmettre un message à v, alors v peut transmettre un message à u. De plus, nous considérons que durant une transmission, les deux liens symétriques sont utilisés (messages d'acquittement notamment). Pour simplifier, nous supposerons dans toute la suite du chapitre que le graphe est non orienté ($G = (V, E)$).

4.1.2.2 Modèle d'interférence binaire

Les interférences peuvent être modélisées de différentes manières. Nous pouvons utiliser le modèle SINR (*Signal-to-Interference-Noise Ratio*) [CXW09] mais cela rend les algorithmes et leurs analyses difficiles. Nous avons choisi de considérer un modèle d'interférence binaire symétrique comme dans [BSS09, GLS07, Wan09]. Nous définissons la zone d'interférence d'une arête $e \in E$, notée $\varepsilon(e)$, par l'ensemble des arêtes interférant avec e. En d'autres termes, si l'arête e et une arête $e' \in \varepsilon(e)$ sont actives simultanément, alors il y a interférence et aucun message n'est correctement envoyé. L'algorithme que nous proposons est valide quel que soit $\varepsilon(e)$, mais dans les calculs, les exemples et les simulations, nous utilisons un modèle d'interférence basé sur les distances (Définition 29).

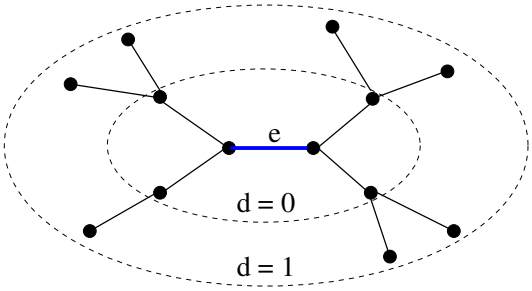


FIG. 4.1 – Zone d'interférence de l'arête $e \in E$ pour le modèle d'interférence à distance $d = 0$ et pour le modèle d'interférence à distance $d = 1$.

Définition 29 (modèle d'interférence à distance d) *Soit $d \geq 0$. Dans le modèle d'interférence à distance d, deux arêtes interfèrent si et seulement si un sommet de la première est à distance (dans G) au plus d d'un sommet de la deuxième arête. Plus précisément, la zone d'interférence d'une arête $e = (u_1, u_2) \in E$ est $\varepsilon(e) = \{(v_1, v_2) \in E, \exists i, j \in \{1, 2\}, d(u_i, v_j) \leq d\}$, avec $d(u, v)$ la distance entre u et v dans G, c'est-à-dire la longueur d'un plus court chemin entre ces deux sommets dans G.*

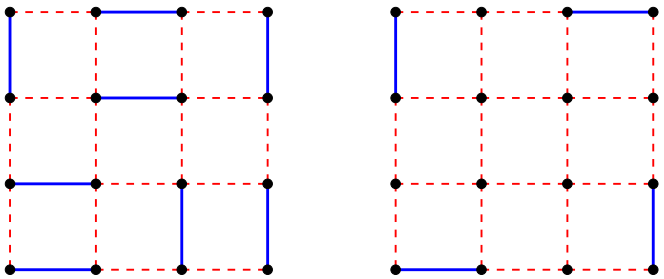
Le modèle d'interférence à distance $d = 0$ est un cas particulier couramment utilisé et connu comme le *primary node interference model* ou le *node exclusive interference model*. Dans ce cas, deux arêtes interfèrent si et seulement si elles sont incidentes. Le modèle d'interférence à distance $d = 1$ est plus réaliste et est connu pour être le modèle d'interférence du 802.11. Dans ce cas, deux sommets peuvent échanger des données seulement si leurs voisinages ne sont pas en train de communiquer. La figure 4.1 représentent deux zones d'interférence d'une arête $e \in E$ pour $d = 0$ et pour $d = 1$.

4.1.2.3 Ensemble valide d'arêtes actives

Un ensemble valide d'arêtes actives est un sous-ensemble d'arêtes pouvant envoyer des messages simultanénement sans aucune interférence. Formellement :

Définition 30 *Etant donné un modèle d'interférence binaire définissant $\varepsilon(e)$ pour tout $e \in E$, un ensemble valide d'arêtes actives $E^* \subseteq E$ est telle que $\forall e, e' \in E^*$, alors $e' \notin \varepsilon(e)$ et $e \notin \varepsilon(e')$.*

Pour le modèle d'interférence à distance $d = 0$ ($d = 1$ respectivement), un ensemble valide d'arêtes actives est un couplage (un couplage induit respectivement) de G. Deux exemples d'ensembles valides d'arêtes actives sont représentés dans la figure 4.2 pour une grille de 16 sommets pour le modèle d'interférence à distance $d = 0$ (figure 4.2(a)) et pour le modèle d'interférence à distance $d = 1$ (figure 4.2(b)).



(a) Couplage de G. (b) Couplage induit de G.

FIG. 4.2 – Deux ensembles valides d'arêtes actives (arêtes pleines bleues) pour une grille G pour le modèle d'interférence à distance $d = 0$ (figure 4.2(a)) et pour le modèle d'interférence à distance $d = 1$ (figure 4.2(b)).

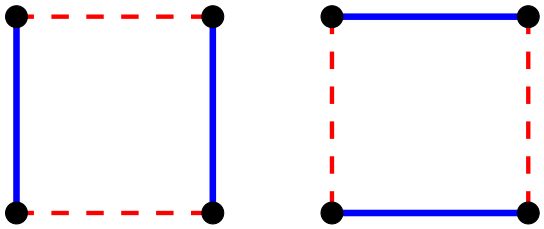
4.1.2.4 Phase de contrôle et phase d'envoi des données

Nous supposons que le temps est divisé en *étapes* (ou *time slots*), toutes de même taille. Les étapes sont représentées par l'ensemble des entiers naturels. Le réseau est synchronisé. Chaque étape $t \geq 1$ est composée de deux différentes phases : une *phase de contrôle* elle-même divisée en *mini-slots* consistant à déterminer un ensemble valide d'arêtes actives E_t^* pour la *phase d'envoi des données* (la seconde phase).

Pour chaque étape $t \geq 1$ et pour chaque arête $e \in E$, nous introduisons la variable $a_t(e)$ telle que $a_t(e) = 1$ si e est active durant la phase d'envoi des données de l'étape t, $a_t(e) = 0$ sinon. Ainsi, un ensemble valide d'arêtes actives à l'étape t vérifie que $\forall e \in E$, $\forall e' \in \varepsilon(e)$, $a_t(e) + a_t(e') \leq 1$.

4.1.2.5 Dynamique du système

Le trafic est supposé *dynamique* et *single-hop*, comme dans [BSS09], c'est-à-dire qu'un message envoyé quitte le réseau juste après. Cependant nous pouvons prendre en compte un trafic *multi-hop* en apportant quelques modifications à l'algorithme distribué que nous proposons. Nous associons à chaque arête $e \in E$, une file d'attente dans laquelle sont stockés les messages. Nous notons $q_t(e)$ le nombre de messages dans la file d'attente de l'arête $e \in E$ au début de l'étape $t \geq 1$. La capacité d'une arête $e \in E$, notée $c(e)$, est le nombre de messages que e peut envoyer durant une étape $t \geq 1$ si e est active. Nous définissons le poids de $e \in E$ à l'étape $t \geq 1$ comme étant la valeur $\lfloor \frac{q_t(e)}{c(e)} \rfloor$. Intuitivement, le poids de e représente le nombre minimum d'étapes pour que le nombre de messages dans la file d'attente associée à e soit strictement plus petit que sa capacité $c(e)$. A chaque étape, de nouveaux messages arrivent sur les arêtes de G selon des processus aléatoires. Ces derniers sont propres à chaque arête $e \in E$ et pas nécessairement connus des concepteurs du réseau. Soit $A_t(e)$ le nombre de messages arrivés sur l'arête $e \in E$ à l'étape $t \geq 1$. Nous supposons



(a) Couplage 1 (b) Couplage 2

FIG. 4.3 – Algorithme d'ordonnancement pour un cycle de quatre sommets avec le modèle d'interférence à distance $d = 0$.

que les messages arrivés sur une arête à l'étape t peuvent être envoyés au plus tôt à l'étape $t+1$. Nous pouvons alors définir la dynamique du système de files d'attente associées aux arêtes.
Dynamique du système : $\forall t \geq 1, \forall e \in E, q_{t+1}(e) = (q_t(e) - a_t(e)c(e))^+ + A_t(e)$ avec $(y)^+ = max(y, 0)$.

4.1.2.6 Exemple d'algorithme d'ordonnancement

Considérons le cycle de quatre sommets de la figure 4.3 et le modèle d'interférence à distance $d = 0$. Nous pouvons activer à chaque étape les arêtes verticales (figure 4.3(a)) ou les arêtes horizontales (figure 4.3(b)). Il est également possible d'activer une unique arête mais nous considérons uniquement les deux ensembles maximaux précédents. Un "bon" algorithme d'ordonnancement des arêtes doit assurer la stabilité du système des quatre files d'attente associées aux quatre arêtes du graphe. La capacité de chacune des arêtes $e \in E$ est $c(e) = 1$ et le nombre moyen d'arrivées de messages est $A_v = \frac{2}{7}$ pour les deux arêtes verticales et $A_h = \frac{4}{7}$ pour les deux arêtes horizontales. Nous pouvons alors concevoir un algorithme d'ordonnancement très simple. En effet, si nous activons les arêtes verticales 5 étapes sur 14 et les arêtes horizontales 9 étapes sur 14, nous assurons la stabilité du système de files d'attente associées aux arêtes car $A_h < \frac{9}{14}$ et $A_v < \frac{5}{14}$.

4.1.3 Travaux existants

Pour le modèle d'interférence à distance $d = 0$, un ensemble valide d'arêtes actives est un couplage de G. Des algorithmes centralisés ont été proposés pour résoudre ce problème aussi bien pour des arrivées stochastiques [Tas97, TE90] que pour des arrivées déterministes [KMP08]. Le problème de trouver un couplage de poids maximum se résout en temps polynomial [LP86].

Pour le modèle d'interférence à distance $d \geq 1$, le problème de trouver un ensemble valide d'arêtes actives maximisant la somme des poids est NP-Complet. Par exemple, le problème de trouver un couplage induit dans un graphe est NP-Complet [SV82, Cam89] et reste NP-Complet même pour des classes de graphes

particulières (pour les graphes 3-réguliers par exemple). Cela correspond au modèle d'interférence à distance $d = 1$. De plus, il est prouvé dans [OFGZ08] que le problème de déterminer un couplage induit maximum dans un graphe ne peut pas être approximé en temps polynomial à un facteur constant près plus petit que $n^{\frac{1}{2}-c}$ pour toute constante $c > 0$ avec n le nombre de sommets du graphe, à moins que P=NP.

Dans [GLS07, BZM06, EAM07, MSZ06], des algorithmes distribués sont décrits mais avec des phases de contrôle croissantes en la taille du réseau. En particulier, [GLS07] présente un algorithme distribué valide quel que soit le modèle d'interférence binaire mais avec une phase de contrôle de durée non constante. Un algorithme distribué avec un nombre constant de mini-slots de contrôle a été proposé (ce nombre dépend de la qualité de l'approximation désirée) ; cependant il est valide uniquement pour le modèle d'interférence à distance $d = 0$. Récemment dans [RSS09], un algorithme aléatoire a été proposé et conjecturé comme étant optimal en termes de débit.

Nous présentons dans la suite deux algorithmes distribués proposés dans [GLS07] (section 4.1.3.1) et dans [BSS09] (section 4.1.3.2). Le nombre de mini-slots de la phase de contrôle de l'algorithme de [GLS07] n'est pas constant alors que l'algorithme décrit dans [BSS09] est valide uniquement pour le modèle d'interférence à distance $d = 0$.

4.1.3.1 Algorithme distribué basé sur les poids des arêtes voisines

La phase de contrôle de l'`Algorithme basé sur les poids des arêtes voisines` proposé dans [GLS07] est composée de T mini-slots. Chaque arête $e \in E$ est initialement indéterminée, c'est-à-dire que e ne sait pas si elle sera active ou inactive durant la phase d'envoi. Chaque arête $e \in E$ choisit une *valeur de backoff* $t(e)$, $1 \leq t(e) \leq T$. L'arête e envoie un message de contrôle lors du mini-slot $t(e)$ si e est encore indéterminée. Pour chaque mini-slot t, $1 \leq t \leq T$, et pour chaque arête $e \in E$ encore indéterminée, nous avons que :

(a) si e reçoit un message de contrôle, alors e devient **inactive** ;
(b) si e ne reçoit pas de message de contrôle et si $t < t(e)$, alors e reste **indéterminée** ;
(c) si e ne reçoit pas de message de contrôle et si $t = t(e)$, alors e devient **active**.

Après la phase de contrôle, l'ensemble des arêtes actives est déterminé. Cet algorithme très simple est valide pour tout modèle d'interférence binaire mais le choix de la valeur de backoff $t(e)$ pour chaque arête $e \in E$, est fonction des poids des arêtes localisées dans $\varepsilon(e)$ et également fonction des poids des arêtes localisées dans la zone d'interférence de chacune des arêtes $e' \in \varepsilon(e)$. Plus précisément $t(e)$ dépend du poids des arêtes appartenant à l'ensemble $S = \{e', e' \in \varepsilon(e)\} \cup \{e'', e'' \in \varepsilon(e'), e' \in \varepsilon(e)\}$. Ainsi à chaque étape $t \geq 1$, chaque arête $e \in E$ doit connaître les poids des arêtes localisées dans son $2d$-voisinage (les arêtes qui sont à distance au plus $2d$ de e) pour le modèle d'interférence à distance d. En définitive, le nombre de mini-slots de la phase de contrôle n'est pas constant. De plus, obtenir ces informations est

un problème aussi difficile que le problème d'ordonnancement distribué des liens en raison des interférences.

4.1.3.2 Algorithme distribué basé sur les chemins augmentants

L'`Algorithme des chemins augmentants` proposé dans [BSS09] a un nombre de mini-slots de contrôle constant mais est spécifique au modèle d'interférence à distance $d = 0$. Il utilise la technique des chemins augmentants qui a été développée dans la théorie des couplages [LP86] pour obtenir des algorithmes centralisés polynomiaux calculant des couplages de poids maximum. Rappelons qu'à chaque étape $t \geq 1$, un ensemble valide d'arêtes actives est un couplage de G. L'idée principale de l'`Algorithme des chemins augmentants` est de calculer, à chaque étape $t+1 \geq 2$, un couplage C_{t+1} de G à partir du couplage C_t de G déterminé à l'étape précédente t. Le poids de C_{t+1} est plus grand ou égal au poids de C_t. Le poids d'un couplage est la somme des poids des arêtes appartenant à ce couplage. Dans [BSS09], il est prouvé que, si $2k+1$ est la longueur maximum des chemins augmentants, alors le nombre de mini-slots de contrôle est $4k+2$ et le couplage réalise $\frac{k}{k+2}$ de la région de capacité.

L'`Algorithme des chemins augmentants` étant distribué, au début de chaque étape $t \geq 1$, chaque sommet $v \in V$ devient *source* avec une probabilité constante p. Une source est un sommet autorisé à débuter un chemin alternant arête active et arête inactive dans le but de trouver un couplage de poids plus grand. Une des limites est que dans [BSS09], il n'est pas expliqué comment calculer analytiquement la valeur de p : une valeur trop petite ne permet pas de trouver rapidement un meilleur couplage car peu de chemins augmentants sont trouvés et une valeur trop grande peut générer trop d'interférence empêchant les chemins augmentants d'être déterminés.

Enfin, il est nécessaire de préciser que l'`Algorithme des chemins augmentants` est spécifique au modèle d'interférence à distance $d = 0$, qui n'est pas le plus réaliste. En effet, la technique des chemins augmentants est propre à la notion de couplage dans les graphes. Rappelons que le problème de déterminer un couplage maximum peut être résolu en temps polynomial avec un algorithme centralisé, mais pour un modèle d'interférence à distance $d \geq 1$, le problème de déterminer un ensemble valide d'arêtes actives de poids maximum est NP-complet.

4.1.4 Contributions

Nous proposons un algorithme distribué d'ordonnancement des liens, `Algorithme Log`, valide quel que soit le modèle d'interférence binaire et avec un nombre de mini-slots de contrôle *indépendant* de la taille du réseau. En effet, ce nombre augmente de manière logarithmique avec le *degré maximum* des nœuds du réseau et ne demande pas de *connaissance explicite* de la taille des files d'attente entre les nœuds. D'un point de vue pratique, cela représente un bon compromis car le degré de chaque nœud est borné en raison de caractéristiques physiques alors que

la taille du réseau peut être arbitrairement grande.

Le nombre de mini-slots de contrôle augmente avec la taille du réseau dans [GLS07] et le modèle d'interférence est contraint d'être le modèle à distance $d = 0$ dans [BSS09]. De plus dans [BSS09, GLS07], l'échange des poids entre certaines arêtes (proches) est nécessaire. Le problème d'acquérir cette information est, à cause des interférences, un problème difficile et demande lui même une phase de contrôle. Dans l'`Algorithme Log`, durant chaque mini-slot de la phase de contrôle, une arête envoie un message composé d'un unique bit ou n'envoie rien. L'idée principale d'`Algorithme Log` est d'utiliser les interférences comme sources d'information. Plus précisément, l'unique préoccupation d'une arête $e \in E$ durant chaque mini-slot est de savoir si une arête $e' \in \varepsilon(e)$ a envoyé un message de contrôle durant ce mini-slot. A partir de cela et de son propre bit, une arête décide si elle devient active, inactive ou si elle retarde sa décision à un autre mini-slot. Comme deux arêtes $e \in E$ et $e' \in \varepsilon(e)$ n'échangent pas leurs poids respectifs durant la phase de contrôle, le nombre et la taille des mini-slots sont donc minimisés.

Une autre idée consiste à associer à chacune des arêtes un poids virtuel (voir section 4.2.1) de telle sorte que deux arêtes interférentes ont des poids virtuels différents (Lemme 24, section 4.2.1). Chaque arête calcule alors un vecteur binaire de contrôle correspondant au poids virtuel ; les bits 1 indiquent les mini-slots dans lesquels l'arête enverra un message de contrôle, si elle n'est pas déjà active ou inactive.

Nous analysons l'`Algorithme Log` dans la section 4.3. Nous prouvons que l'ensemble d'arêtes actives est toujours maximal (Théorème 21, section 4.3.1) et nous discutons les valeurs des différents paramètres (section 4.3.3 et section 4.3.4). Nous étudions ensuite les différentes conditions assurant la stabilité du système de files d'attente associées aux arêtes du graphe (section 4.3.5).

Enfin, nous décrivons dans la section 4.4 les performances d'`Algorithme Log` via des simulations pour différents graphes : chemins, grilles et graphes aléatoires. Nous comparons également l'`Algorithme Log` avec l'`Algorithme des chemins augmentants` en utilisant les paramètres de simulation décrits dans [BSS09].

4.2 Algorithme Log

Soit une étape $t \geq 1$. Nous notons $G_t = (V, E_t)$ le graphe induit par les arêtes de poids strictement positifs. Précisément : $E_t = \{e \in E, \lfloor \frac{q_t(e)}{c(e)} \rfloor \geq 1\}$. L'ensemble d'arêtes actives E_t^* déterminé par l'`Algorithme Log` est composé uniquement d'arêtes de E_t. Dans la suite, le modèle d'interférence binaire est quelconque sauf mention contraire et l'`Algorithme Log` est décrit pour une étape $t \geq 1$ quelconque.

Avant de décrire formellement l'`Algorithme Log` dans la section 4.2.2, nous décrivons ses ingrédients principaux : la notion de poids virtuel et de vecteur de contrôle d'une arête (section 4.2.1).

4.2.1 Poids virtuels et vecteur de contrôle

L'idée consiste à diviser les poids des arêtes de E_t en un petit nombre K d'intervalles (classes) disjoints $I_0, \ldots, I_{K-1}$ avec la contrainte que la plus grande classe I_{K-1} contient toutes les valeurs plus grandes ou égales à une valeur L : $I_{K-1} = [L, \infty[$. Comme K est petit, des arêtes d'une même zone d'interférence peuvent appartenir au même intervalle. Pour les différencier, nous attribuons une couleur (un entier) $\gamma(e)$ à chaque arête $e \in E$ telle que deux arêtes qui interfèrent ont des couleurs différentes. Soit C le nombre total de couleurs. Par exemple si G est un chemin d'au moins $d+2$ arêtes et si nous considérons le modèle d'interférence à distance d, nous pouvons utiliser seulement $d+2$ couleurs en donnant les valeurs $1, 2, \ldots, d+2, 1, 2, \ldots, d+2, \ldots$ aux arêtes. Dans un souci d'équité, nous permutons les couleurs à chaque étape. Pour cela, nous attribuons à chaque arête $e \in E$ et à chaque étape $t \geq 1$, la valeur $\gamma_t(e) = (\gamma(e) + t - 2)_{modC} + 1$.

Propriété 6 *$\forall e \in E$, $\forall e' \in \varepsilon(e)$, $\forall t \geq 1$, nous avons $\gamma_t(e) \neq \gamma_t(e')$.*

Par exemple avec $C = 3$ couleurs, une arête $e \in E$ de couleur $\gamma(e) = 1$ aura une valeur $\gamma_t(e) = 1$ à l'étape 1, une valeur $\gamma_t(e) = 2$ à l'étape 2, une valeur $\gamma_t(e) = 3$ à l'étape 3, une valeur $\gamma_t(e) = 1$ à l'étape 4, une valeur $\gamma_t(e) = 2$ à l'étape 5 et ainsi de suite.

Nous pouvons définir le poids virtuel $q'_t(e)$ pour toute arête $e \in E$ comme suit.

Définition 31 (poids virtuel) *Soit la fonction $f : [1, \infty[\rightarrow \{0, \ldots, K-1\}$ définie par $f(x) = i$ si $x \in I_i$ pour $i = 0, 1, \ldots, K-1$.*
Pour tout $e \in E_t$, le poids virtuel de e est :

$$q'_t(e) = Cf\left(\lfloor \frac{q_t(e)}{c(e)} \rfloor\right) + \gamma_t(e) \tag{4.1}$$

Pour tout $e \in E \setminus E_t$ ($q_t(e) < c(e)$), le poids virtuel de e est $q'_t(e) = 0$.

Propriété 7 *$\forall t \geq 1, \forall e \in E$, nous avons $0 \leq q'_t(e) \leq CK$.*

Les choix des constantes C, K et L sont discutés dans la section 4.3.3 et dans la section 4.3.4. Nous supposons dans la suite que ces valeurs sont données et que C est suffisamment grand pour garantir les propriétés décrites précédemment.

L'introduction des poids virtuels est triplement motivée. Premièrement, et surtout, les poids virtuels vérifient la propriété que deux liens qui interfèrent ont des poids virtuels différents (Lemme 24). Cette propriété est l'ingrédient majeur d'`Algorithme Log`. En effet, il est certain qu'à chaque étape, l'`Algorithme Log` détermine un ensemble valide d'arêtes actives maximal (Théorème 21, section 4.3). Deuxièmement, la normalisation décrite dans la section 4.1.2 (c'est-à-dire le fait que $q_t(e)$ est divisé par $c(e)$) assure qu'`Algorithme Log` ne favorise pas les liens de grandes capacités. Enfin, `Algorithme Log` utilise l'ordre induit par les poids virtuels pour déterminer les arêtes actives ; une arête avec un grand poids virtuel et en particulier avec une file d'attente chargée aura plus de chances d'être choisie.

Lemme 24 *Soient $e, e' \in E_t$ telles que $e' \in \varepsilon(e)$ (et $e \in \varepsilon(e')$). Alors, $q'_t(e) \neq q'_t(e')$.*

Preuve : Si $\lfloor \frac{q_t(e)}{c(e)} \rfloor$ et $\lfloor \frac{q_t(e')}{c(e')} \rfloor$ appartiennent à deux intervalles différents, c'est-à-dire si $f(\lfloor \frac{q_t(e)}{c(e)} \rfloor) \neq f(\lfloor \frac{q_t(e')}{c(e')} \rfloor)$, alors $|C(f(\lfloor \frac{q_t(e)}{c(e)} \rfloor) - f(\lfloor \frac{q_t(e')}{c(e')} \rfloor))| \geq C$ et comme $1 \leq \gamma_t(e), \gamma_t(e') \leq C$, nous avons alors $q'_t(e) \neq q'_t(e')$. Sinon, si $f(\lfloor \frac{q_t(e)}{c(e)} \rfloor) = f(\lfloor \frac{q_t(e')}{c(e')} \rfloor)$, alors par la Propriété 6, nous avons $\gamma_t(e) \neq \gamma_t(e')$ et donc $q'_t(e) \neq q'_t(e')$. □

Nous associons à chaque arête $e \in E$ un vecteur de contrôle $v_{t,e}$ comme suit.

Définition 32 (vecteur de contrôle) *Le* vecteur de contrôle $v_{t,e}$ *d'une arête $e \in E$ est $v_{t,e} = (v_{t,e}(1), \ldots, v_{t,e}(T))$ avec $v_{t,e}(i)$, $1 \leq i \leq T$, correspondant au i^e bit de $q'_t(e)$. La taille du vecteur est $T = \lceil log_2(CK) + 1 \rceil$.*

Le tableau 4.1 représente tous les poids virtuels $q'_t(e)$ possibles, ainsi que les vecteurs de contrôle $v_{t,e}$ correspondant, pour chaque paire possible $(\lfloor \frac{q_t(e)}{c(e)} \rfloor, \gamma_t(e))$ d'une arête $e \in E$ avec les paramètres $C = 3$, $K = 5$ et $L = 5$. Nous aurions pu choisir $L = 101$ au lieu de $L = 5$ et prendre comme intervalles $I_0 = [1, 10]$, $I_1 =]10, 30]$, $I_2 =]30, 60]$, $I_3 =]60, 100]$ et $I_4 =]100, \infty[$ ou tout autre division des poids respectant les contraintes décrites précédemment. La valeur $C = 3$ correspond par exemple au nombre minimum de couleurs pour un cycle composé d'un nombre impair de sommets avec le modèle d'interférence à distance $d = 0$. La figure 4.4 est un cycle composé de 10 arêtes. Soient le modèle d'interférence à distance $d = 0$ et les paramètres $C = 3$, $K = 5$ et $L = 5$. Les valeurs $\lfloor \frac{q_t(e)}{c(e)} \rfloor$, $\gamma_t(e)$ et $q'_t(e)$ sont représentées dans la figure 4.4(a), dans la figure 4.4(b) et dans la figure 4.4(c), respectivement. Les quatre bits des vecteurs de contrôle se trouvent dans les figures 4.4(d), 4.4(e), 4.4(f) et 4.4(g).

4.2.2 Algorithme distribué

Etant donnés un réseau représenté par le graphe $G = (V, E)$, un modèle d'interférence binaire définissant $\varepsilon(e)$ pour tout $e \in E$, les constantes K, L et C respectant les contraintes précédemment décrites, l'`Algorithme Log` détermine un ensemble d'arêtes actives maximal à chaque étape $t \geq 1$. Nous décrivons la phase de contrôle d'`Algorithme Log` d'une étape $t \geq 1$. Avant le début de cette phase, chaque arête $e \in E$ est indéterminée, c'est-à-dire que nous ne savons pas si e sera active ou inactive durant la phase d'envoi. La phase de contrôle est divisée en α *sous-phases* composées chacune de mini-slots de contrôle. Après chaque mini-slot, une arête e peut être active, inactive ou encore indéterminée. Nous expliquons tout d'abord la sous-phase 1 (section 4.2.2.1) avant de motiver et de décrire les sous-phases $2, \ldots, \alpha$ (section 4.2.2.2). Nous verrons dans la section 4.3.1 comment déterminer la valeur de α.

4.2.2.1 Sous-phase 1 de la phase de contrôle

La sous-phase 1 est composée de $T = \lceil log_2(CK) + 1 \rceil$ mini-slots. Avant le mini-slot 1, chaque arête est indéterminée. Pendant un mini-slot i, $1 \leq i \leq T$, $e \in E$ envoie

TAB. 4.1 – $(q'_t(e), v_{t,e})$ pour chaque paire possible $(\lfloor \frac{q_t(e)}{c(e)} \rfloor, \gamma_t(e))$ pour l'`Algorithme Log` avec les paramètres $C = 3$, $K = 5$ et $L = 5$.

				Vecteur de contrôle : $v = v_{t,e}$			
$\lfloor \frac{q_t(e)}{c(e)} \rfloor$	$f(\lfloor \frac{q_t(e)}{c(e)} \rfloor)$	$\gamma_t(e)$	$q'_t(e)$	$v(1)$	$v(2)$	$v(3)$	$v(4)$
0	non défini	$\in \{1,2,3\}$	0	0	0	0	0
1	0	1	1	0	0	0	1
1	0	2	2	0	0	1	0
1	0	3	3	0	0	1	1
2	1	1	4	0	1	0	0
2	1	2	5	0	1	0	1
2	1	3	6	0	1	1	0
3	2	1	7	0	1	1	1
3	2	2	8	1	0	0	0
3	2	3	9	1	0	0	1
4	3	1	10	1	0	1	0
4	3	2	11	1	0	1	1
4	3	3	12	1	1	0	0
≥ 5	4	1	13	1	1	0	1
≥ 5	4	2	14	1	1	1	0
≥ 5	4	3	15	1	1	1	1

un message de contrôle si et seulement si e est encore indéterminée et $v_{t,e}(i) = 1$. Si e est encore indéterminée au début du mini-slot i, $1 \leq i \leq T$, trois différents cas sont possibles (voir lignes 4 à 14 de l'Algorithme 6) :

(a) si e envoie un message de contrôle et n'en reçoit pas un d'une arête de $\varepsilon(e)$, alors e devient **active** ;
(b) si e reçoit un message de contrôle (venant d'une arête de $\varepsilon(e)$) et n'en envoie pas un, alors e devient **inactive** ;
(c) sinon e reste **indéterminée**.

A la fin de la sous-phase 1, nous obtenons un ensemble valide d'arêtes actives (un couplage de G pour le modèle d'interférence à distance $d = 0$ par exemple). De plus, à la fin de la sous-phase 1, une arête $e \in E_t$ est soit active soit inactive (Lemme 25).

Lemme 25 *A la fin de la sous-phase 1 d'*`Algorithme Log`*, il n'y a pas d'arêtes indéterminées appartenant à* E_t.

Preuve : Le Lemme 24 garantit que $q'_t(e) \neq q'_t(e')$ pour tout $e, e' \in E_t$ telle que $e' \in \varepsilon(e)$. Supposons que e est indéterminée à la fin de la sous-phase 1 d'`Algorithme Log`. Durant le dernier mini-slot où e a envoyé un message de contrôle, elle est restée

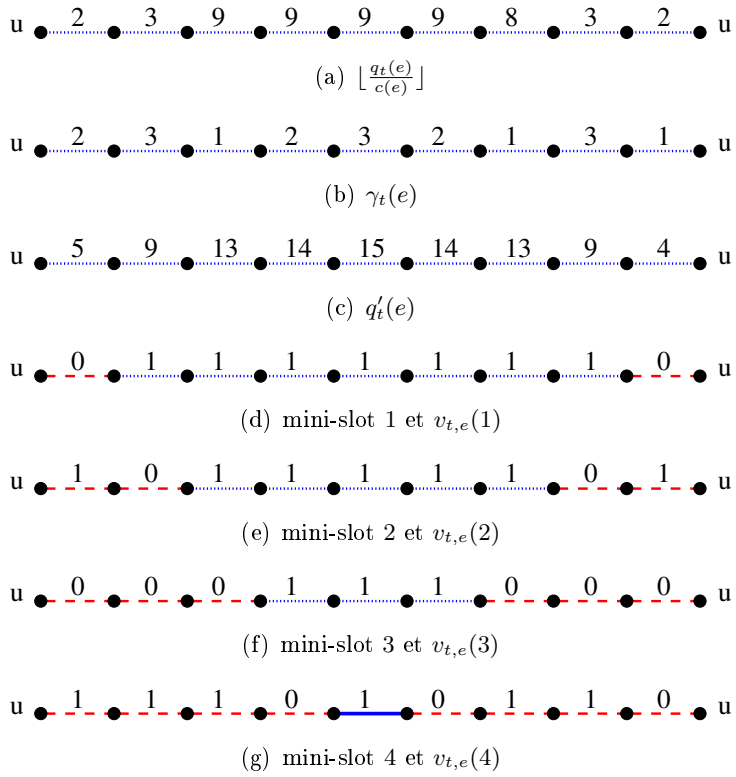


FIG. 4.4 – Sous-phase 1 d'`Algorithme Log` pour un cycle de 9 arêtes avec le modèle d'interférence à distance $d = 0$ et les paramètres $C = 3$, $K = 5$ et $L = 5$.

indéterminée si une autre arête $e' \in \varepsilon(e)$ a envoyé un message durant ce mini-slot. Mais comme $q'_t(e) \neq q'_t(e')$, il existe un mini-slot antérieur où l'arête de plus grand poids a envoyé un message de contrôle et l'autre non ; mais l'arête de plus petit poids virtuel serait devenue inactive à ce mini-slot, une contradiction. □

Définition 33 (arête maximum local) *Une arête $e \in E_t$ est une arête maximum local à l'étape $t \geq 1$, si $\forall e' \in \varepsilon(e)$, alors $q'_t(e') < q'_t(e)$.*

Nous déduisons du Lemme 25 que :

Corollaire 9 *Une arête maximum local est active à la fin de la sous-phase 1 d'*`Algorithme Log`*.*

Les arêtes ayant un gros poids d'un point de vue local appartiennent à l'ensemble des arêtes actives. Ainsi, les files d'attente associées à ces arêtes vont diminuer. En revanche après la sous-phase 1, il est possible que cet ensemble d'arêtes actives ne soit pas maximal (c'est-à-dire que des arêtes peuvent être ajoutées sans en enlever d'autres). La figure 4.4 décrit la sous-phase 1 d'`Algorithme Log` pour un cycle composé de 10 arêtes avec le modèle d'interférence à distance $d = 0$ et les paramètres $C = 3$, $K = 5$ et $L = 5$. Lors du mini-slot 1, toutes les arêtes envoient un message

Algorithme 6 : `Algorithme Log` pour une arête $e \in E$

Précondition : $q_t(e)$, $\gamma_t(e)$.
Postcondition : retourne *active* ou *inactive*

1: e calcule $q'_t(e)$ et $v_{t,e}$
2: $s(e) = 0$
3: **pour** $j = 1, \ldots, \alpha$ **faire**
4: **pour** $i = 1, \ldots, T$ **faire**
5: **si** $s(e) = 0$ et $v_{t,e}(i) = 1$ **alors**
6: e envoie un message (aux arêtes de $\varepsilon(e)$)
7: **si** e ne reçoit pas de message **alors**
8: $s(e) = 1$
9: **si** $s = 0$, $v_{t,e}(i) = 0$ et e reçoit un message **alors**
10: $s(e) = -1$
11: **si** $s(e) = 1$ **alors**
12: e envoie un message (aux arêtes de $\varepsilon(e)$)
13: **si** $s(e) = -1$ et e ne reçoit pas de message **alors**
14: $s(e) = 0$
15: **si** $s(e) = 1$ **alors**
16: retourner *active*
17: **sinon**
18: retourner *inactive*

de contrôle sauf les deux arêtes incidentes au sommet u (figure 4.4(d)). Ainsi, toutes les arêtes restent indéterminées sauf les deux arêtes incidentes à u qui deviennent inactives. Lors du mini-slot 2, deux arêtes deviennent inactives et six restent indéterminées (figure 4.4(e)). Après le mini-slot 3, les arêtes sont toutes inactives saut trois qui sont indéterminées (figure 4.4(f)). Lors du mini-slot 4, l'arête centrale envoie un message de contrôle sans en recevoir un : cette dernière devient active et les deux autres deviennent inactives (figure 4.4(g)). Pour un tel graphe G, cela représente le pire des cas en termes de cardinalité de l'ensemble des arêtes actives formant un couplage de G. En effet, après la sous-phase 1, seule l'arête centrale est active. Une possibilité est d'ajouter des mini-slots pour obtenir plus d'arêtes actives en utilisant un processus aléatoire. Une autre solution consiste à répéter $\alpha - 1$ fois cette sous-phase jusqu'à obtenir un ensemble maximal. Nous décrivons cette dernière solution dans la section 4.2.2.2 et déterminons la valeur de α pour garantir la maximalité (Théorème 21, section 4.3.1).

4.2.2.2 Autres sous-phases de la phase de contrôle

Comme décrit précédemment, il est possible que l'ensemble des arêtes actives ne soit pas maximal après la sous-phase 1 (en considérant G_t). Pour pallier ce problème, nous répétons $\alpha - 1$ fois la sous-phase 1 avec seulement les arêtes inactives qui n'ont pas d'arêtes actives dans leurs zones d'interférence respectives. Rappelons qu'après la sous-phase 1, chaque arête $e \in E_t$ est soit active soit inactive (Lemme 25). Nous

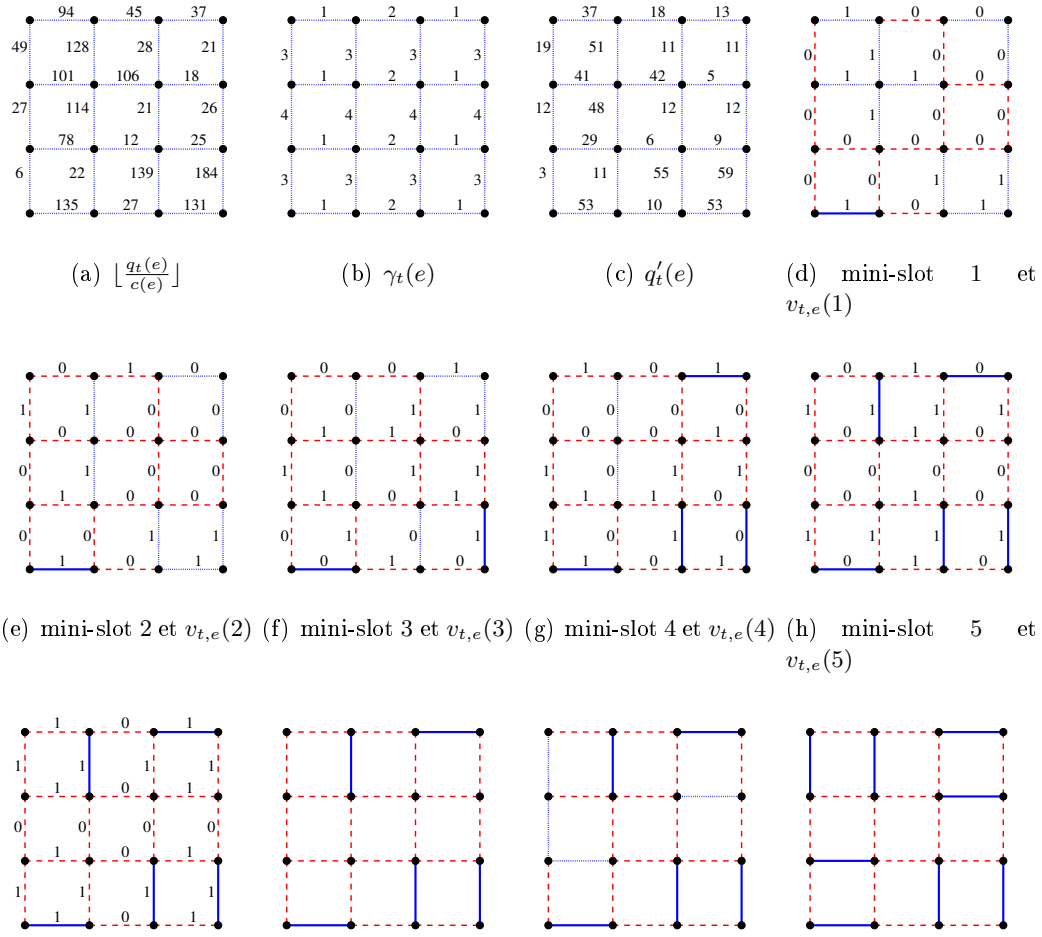
ajoutons alors un mini-slot de réinitialisation après chaque sous-phase j, $1 \leq j \leq \alpha - 1$, dans lequel chaque arête active envoie un message de contrôle et les arêtes inactives redeviennent indéterminées si elles ne reçoivent pas de message (lignes 15 à 20 de l'Algorithme 6). La prochaine sous-phase concernera donc uniquement ces arêtes indéterminées. Le Théorème 21 de la section 4.3.1 montre que si nous choisissons $\alpha = T$, alors à la fin de la phase de contrôle d'`Algorithme Log`, l'ensemble d'arêtes actives est maximal en considérant G_t.

L'`Algorithme Log` est alors défini par les α sous-phases décrites précédemment. L'Algorithme 6 décrit formellement l'algorithme utilisé par chacune des arêtes : $s(e) = 1$ si e est active, $s(e) = 0$ si e est indéterminée et $s(e) = -1$ si e est inactive.

Nous présentons dans la suite deux applications d'`Algorithme Log` : pour une grille avec le modèle d'interférence à distance $d = 0$ (section 4.2.2.3) et pour un graphe aléatoire avec le modèle d'interférence à distance $d = 1$ (section 4.2.2.4). Nous décrivons le modèle aléatoire que nous utilisons dans la section 4.4.2.3.

4.2.2.3 Grille avec le modèle d'interférence à distance d=0

Soit $G = (V, E)$ une grille composée de $|V| = 16$ sommets et de $|E| = 24$ arêtes (figure 4.5). Le modèle d'interférence à distance $d = 0$ assure qu'un ensemble valide d'arêtes actives est un couplage de G. Dans notre exemple, le plus petit C respectant les contraintes décrites dans la section 4.2.1 est $C = 4$ (voir la figure 4.5(b) pour un exemple de coloration optimale). Nous choisissons $K = 15$ et $L = 140$. Soit l'étape $t \geq 1$. Les valeurs $\lfloor \frac{q_t(e)}{c(e)} \rfloor$, $\gamma_t(e)$ et $q'_t(e)$ sont représentées dans la figure 4.5(a), dans la figure 4.5(b) et dans la figure 4.5(c) pour chaque arête $e \in E$, respectivement. Par exemple pour l'arête e de poids $q_t(e) = 94$ et de couleur $\gamma_t(e) = 1$, nous avons $q'_t(e) = 37$ en utilisant l'équation 4.1 de la Définition 31. Le vecteur de contrôle est alors $v_{t,e}(e) = (1, 0, 0, 1, 0, 1)$. Décrivons maintenant la sous-phase 1 d'`Algorithme Log` composée de $T = \lceil log_2(CK) + 1 \rceil = 6$ mini-slots. Avant le mini-slot 1, chaque arête est indéterminée. La figure 4.5(d) représente l'état des arêtes après le mini-slot 1. Par exemple l'arête ayant $q'_t(e) = 53$ devient active. De plus, il y a trois composantes connexes d'arêtes indéterminées composées de 5, 3 et 2 arêtes, respectivement. Après le mini-slot 2, il y a trois composantes connexes d'arêtes indéterminées composées de 2, 3 et 2 arêtes, respectivement (figure 4.5(e)). Après le mini-slot 3, une deuxième arête devient active et il y a trois composantes connexes d'arêtes indéterminées composées de 2, 1 et 2 arêtes, respectivement (figure 4.5(f)). Après le mini-slot 4, il y a deux arêtes actives supplémentaires et plus qu'une composante connexe composée de 2 arêtes. Après le mini-slot 5, une arête devient active et il n'y a plus d'arêtes indéterminées (figure 4.5(g)). Ainsi après $T = 6$ mini-slots (c'est-à-dire après la sous-phase 1 d'`Algorithme Log`), un ensemble valide d'arêtes actives est déterminé. En revanche, il est facile de vérifier que ce couplage n'est pas maximal. Au début de la sous-phase 2, les arêtes inactives sans arêtes actives dans leurs zones d'interférence redeviennent indéterminées (cela est possible avec un mini-slot). Après la sous-phase 2, un couplage maximal est déterminé. Le Théorème 21 de la section 4.3.1 garantit que $\alpha = T$ sous-phases sont suffisantes pour



(a) $\lfloor \frac{q_t(e)}{c(e)} \rfloor$ (b) $\gamma_t(e)$ (c) $q'_t(e)$ (d) mini-slot 1 et $v_{t,e}(1)$

(e) mini-slot 2 et $v_{t,e}(2)$ (f) mini-slot 3 et $v_{t,e}(3)$ (g) mini-slot 4 et $v_{t,e}(4)$ (h) mini-slot 5 et $v_{t,e}(5)$

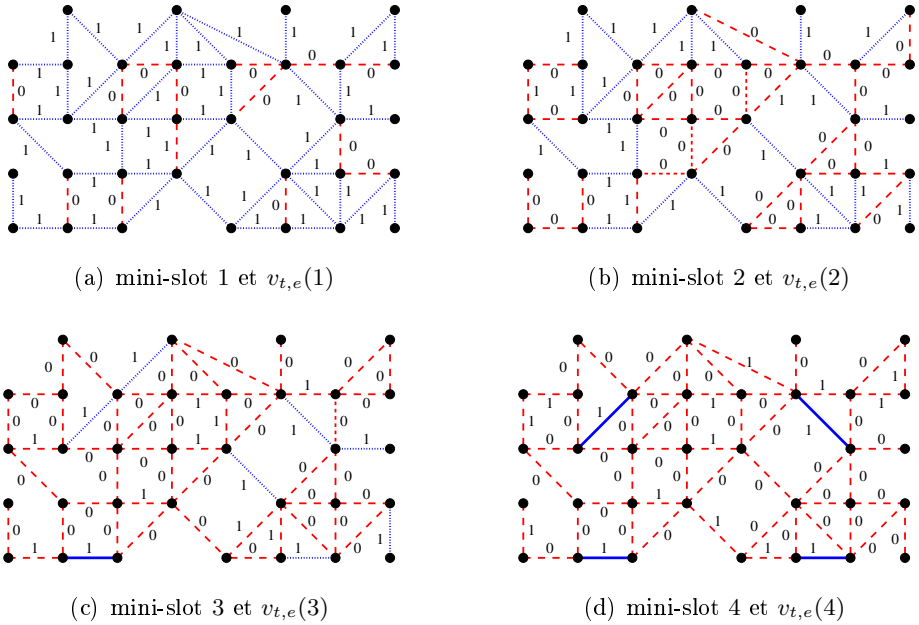
(i) mini-slot 6 et $v_{t,e}(6)$ (j) Après la sous-phase 1 (k) Avant la sous-phase 2 (l) Après la sous-phase 2

FIG. 4.5 – `Algorithme Log` pour une grille avec le modèle d'interférence à distance $d = 0$ et les paramètres $C = 4$, $K = 15$ et $L = 140$.

obtenir cette propriété de maximalité.

4.2.2.4 Graphe aléatoire avec le modèle d'interférence à distance d=1

Considérons le graphe $G = (V, E)$ composé de $|V| = 33$ sommets et de $|E| = 56$ arêtes (figure 4.6). G a été généré selon le processus décrit dans la section 4.4.2.3. Le modèle d'interférence à distance $d = 1$ assure qu'un ensemble valide d'arêtes actives forme un couplage induit de G. La figure 4.6 montre les quatre premiers mini-slots de la sous-phase 1 d'`Algorithme Log`. Nous obtenons après le mini-slot 4 un couplage induit maximal et donc l'ensemble des arêtes actives.



(a) mini-slot 1 et $v_{t,e}(1)$ (b) mini-slot 2 et $v_{t,e}(2)$

(c) mini-slot 3 et $v_{t,e}(3)$ (d) mini-slot 4 et $v_{t,e}(4)$

FIG. 4.6 – Mini-slots 1, 2, 3 et 4 de la sous-phase 1 d'`Algorithme Log` pour un graphe aléatoire avec le modèle d'interférence à distance $d = 1$.

4.3 Analyse

Dans cette section, nous fixons tout d'abord la valeur α pour assurer qu'à chaque étape $t \geq 1$, l'ensemble des arêtes actives est maximal (section 4.3.1). Nous en déduisons, dans la section 4.3.2, le nombre de mini-slots de contrôle d'`Algorithme Log`. Enfin, nous décrivons comment déterminer les différents paramètres C, K et L (section 4.3.3 et section 4.3.4).

4.3.1 Maximalité de l'ensemble des arêtes actives

Pour toute étape $t \geq 1$, nous prouvons dans le Théorème 21 qu'après la phase de contrôle d'`Algorithme Log`, l'ensemble des arêtes actives est toujours maximal (en considérant G_t) si nous choisissons $\alpha = T = \lceil log_2(CK) + 1 \rceil$.

Théorème 21 *Si nous choisissons* $\alpha = T = \lceil log_2(CK) + 1 \rceil$ *dans l'*`Algorithme Log`*, alors* $\forall e \in E_t$*, il existe (au moins) une arête* $e' \in \varepsilon(e) \cup \{e\}$ *telle que* e' *est active après la phase de contrôle d'*`Algorithme Log`*.*

Preuve : Soit $T = \lceil log_2(CK) + 1 \rceil$. Soit e_1 une arête quelconque de E_t. Soit e_1 est active à la fin de la sous-phase 1 d'`Algorithme Log` et le théorème est prouvé. Sinon e_1 est inactive à cause d'une arête $e_2 \in \varepsilon(e_1)$ (avec $q'_t(e_2) > q'_t(e_1)$) qui a envoyé un message de contrôle à un certain mini-slot t_1. Soit e_2 est active et le théorème est prouvé, soit e_2 est inactive à cause d'une arête $e_3 \in \varepsilon(e_2)$ qui a envoyé un message

de contrôle à un certain mini-slot $t_2 > t_1$ et ainsi de suite. Soit k le plus grand indice d'une séquence d'arêtes inactives $e_1, e_2, \ldots, e_k$ telle que l'arête e_i $(1 \leq i \leq k)$ est inactive à cause de $e_{i+1} \in \varepsilon(e_i)$ qui a envoyé un message de contrôle au mini-slot t_i et l'arête e_{k+1} est active. Comme $T \geq t_k > \ldots > t_i > t_{i-1} > \ldots > t_1 \geq 1$, nous avons $k \leq T$.

Nous prouvons maintenant par induction que, à la fin de la sous-phase j d'`Algorithme Log`, la plus longue séquence d'arêtes $e_1, e_2, \ldots, e_{k_j}$ telle que e_i $(1 \leq i \leq k_j)$ est devenue inactive à cause de $e_{i+1} \in \varepsilon(e_i)$ et l'arête e_{k_j+1} est active, vérifie $k_j \leq T - j + 1$. Comme nous l'avons vu, cela est vrai pour $j = 1$. Supposons maintenant que cela est vrai jusqu'à $j-1$. Remarquons que e_{k_j+1} n'était pas active à la fin de la sous-phase $j-1$, sinon e_{k_j} aurait été définitivement inactive à la fin de la sous-phase $j-1$ et n'aurait pas participé à la sous-phase j . Comme les arêtes $e_1, e_2, \ldots, e_{k_j}$ sont aussi inactives à la fin de la sous-phase $j-1$, nous avons une séquence d'arêtes inactives de taille $k_j + 1$ et par hypothèse la taille de cette séquence vérifie $k_j + 1 \leq T - (j-1) + 1$ et donc $k_j \leq T - j + 1$. Ainsi, pour un certain $j_0 \leq T$, $k_{j_0} \leq 1$ impliquant qu'à la fin de la sous-phase j_0 toute arête e est soit active, soit inactive à cause d'une arête active $e' \in \varepsilon(e) \cup \{e\}$. □

4.3.2 Nombre de messages de contrôle

Nous donnons dans le Théorème 22 le nombre de mini-slots de la phase de contrôle d'`Algorithme Log` en fonction des paramètres K et C.

Théorème 22 *Le nombre de mini-slots de la phase de contrôle d'*`Algorithme Log` *est* $T_{log} = \lceil log_2(CK) + 1 \rceil^2 + \lceil log_2(CK) + 1 \rceil - 1$.

Preuve : La sous-phase 1 d'`Algorithme Log` est composée de $T = \lceil log_2(CK) + 1 \rceil$ mini-slots. Le Théorème 21 garantit qu'avec $\alpha = T$ sous-phases, l'ensemble des arêtes actives est maximal. De plus, à la fin de la sous-phase i, $1 \leq i \leq T-1$, un mini-slot de réinitialisation est nécessaire. En effet, chaque arête active envoie un message de contrôle et un certain nombre d'arêtes inactives redeviennent indéterminées si elles n'ont pas reçu de message. $T-1$ mini-slots supplémentaires sont ajoutés. Donc $T_{log} = T^2 + T - 1$. □

4.3.3 Calcul du nombre C de couleurs

Dans le but d'avoir un nombre petit de mini-slots T_{log}, nous devons choisir une petite valeur pour la constante C tout en assurant que $\forall e \in E$, $\forall e' \in \varepsilon(e)$, alors $\gamma(e) \neq \gamma(e')$ et $1 \leq \gamma(e), \gamma(e') \leq C$. Rappelons que si $\gamma(e) \neq \gamma(e')$, alors $\forall t \geq 1$, nous avons $\gamma_t(e) \neq \gamma_t(e')$.

Etant donné un graphe $G = (V, E)$, le problème consiste à associer un entier (une couleur) $\gamma(e)$ à chacune des arêtes $e \in E$ tels que les deux entiers associés à deux arêtes $e, e' \in E$ avec $e \in \varepsilon(e')$ soient différents, minimisant le nombre total d'entiers (couleurs) utilisés. Dans notre problème l'entier (la couleur) associé à l'arête e correspond à $\gamma(e)$ et le nombre total minmum de couleurs correspond au plus petit

C respectant les contraintes précédentes. Il existe un algorithme glouton pour obtenir $2\Delta(G)^{d+1}$ différents entiers (couleurs) en temps linéaire [Die97] avec $\Delta(G)$ le degré maximum dans G pour le modèle d'interférence à distance d.

Dans le cas particulier du modèle d'interférence à distance $d = 0$, le problème de minimiser C revient à calculer l'*indice chromatique du graphe* (*Edge Chromatic Number*), noté $\chi'(G)$, de G (voir [FW77]). Le Théorème 23 prouve une borne meilleure pour ce nombre minimum de couleurs. Une preuve de ce théorème peut être trouvée dans [FW77] et une preuve constructive dans [MG92]. Enfin, il existe un algorithme polynomial garantissant une (+1)-approximation pour ce problème particulier. Plus précisément, cet algorithme calcule une coloration des arêtes avec au plus $\Delta(G)+1$ couleurs et, dans le pire des cas, la solution optimale est composée de $\Delta(G)$ couleurs.

Théorème 23 (Vizing, [FW77, MG92]) *Etant donné un graphe $G = (V, E)$, $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$ avec $\Delta(G)$ le degré maximum dans G.*

De plus, si G est biparti, alors $\chi'(G) = \Delta(G)$, et en particulier si G est une grille, nous avons $\chi'(G) = 4$.

4.3.4 Choix des constantes K et L

Si le nombre T_{log} de mini-slots et le nombre de couleurs C sont constants et fixés, il est possible de calculer la valeur maximum de K respectant cette contrainte. Plus précisément, nous choisissons la plus grande valeur K telle que $\lceil log_2(CK) + 1\rceil^2 + \lceil log_2(CK) + 1\rceil - 1 \leq T_{log}$. De plus, nous choisissons L suffisamment grand. Rappelons que le nombre de mini-slots T_{log} est constant en L.

4.3.5 Stabilité

Nous analysons dans cette section la stabilité du système de files d'attente avec l'`Algorithme Log`. Comme l'`Algorithme Log` détermine à chaque étape un ensemble d'arêtes actives maximal, il est alors assuré d'obtenir une fraction de la région de capacité pour tout graphe représentant le réseau sans-fil [CKLS08]. Cependant, la région de capacité pour l'`Algorithme Log` est plus grande que cette borne.

Rappelons que $A_t(e)$ est le nombre de messages arrivant sur l'arête e à l'étape t. Rappelons également que $q_t(e)$ est le nombre de messages dans la file d'attente associée à l'arête e au début de l'étape t.

Nous supposons que les arrivées sur l'arête $e \in E$ durant les étapes $sC+1, sC+2, \ldots, (s+1)C$ $(s = 0, 1, \ldots)$ sont prises en compte à la fin de l'étape $(s+1)C$. Soit $B_s(e) := \sum_{t=sC+1}^{(s+1)C} A_t(e)$ le nombre total de messages arrivés sur l'arête $e \in E$ durant les étapes $sC+1, sC+2, \ldots, (s+1)C$. Nous avons :

$$q_{(s+1)C}(e) = q_{sC+1}(e) - x_{sC+1}(e) + B_s(e), \quad e \in E,$$

avec $x_{sC}(e) \in [c(e), q_{sC+1}(e)]$ le nombre de messages transmis via l'arête e pendant les étapes $sC+1, sC+2, \ldots, (s+1)C$. Nous avons $x_{sC}(e) \geq c(e)$ car `Algorithme`

`Log` active une arête e si et seulement si le nombre de messages dans la file d'attente associée est au moins égal à sa capacité $c(e)$ (voir la section 4.2.1).

Hypothèse 1 *Pour toute arête $e \in E$, $\{B_s(e), s \geq 0\}$ est une séquence indépendante et identiquement distribuée de variables aléatoires. De plus $\{B_s(e), s \geq 0\}$, $e \in E$, sont des séquences mutuellement indépendantes.*

L'Hypothèse 1 et la définition des poids virtuels de la section 4.2.1 impliquent que $\mathbf{X} := \{((q_{sC}(e), \gamma_{sC}(e)), e \in E), s \geq 0\}$ est un processus de Markov d'espace d'états $\mathcal{N}^{|E|} \times \{1, \ldots, C\}$, avec C le nombre de couleurs respectant les contraintes liées aux interférences. Nous supposons que cette chaîne de Markov est irréductible. En particulier, si $P(A_t(e) = k) > 0$ pour tout k, nous obtenons l'irréductibilité. La stabilité d'`Algorithme Log` est définie par la stabilité de la chaîne de Markov $\mathbf{X}$.

Soit $\{\mathbf{I}_j\}_j$ l'ensemble de tous les ordonnancements valides. Un ordonnancement $\mathbf{I}_j = (I_j(e), e \in E)$ est un vecteur binaire avec $I_j(e) = 1$ si e est active et $I_j(e) = 0$ sinon. Un ordonnancement est valide s'il ne produit pas d'interférence entre les différentes arêtes actives.

Nous définissons la région de capacité (*capacity region*) $\mathcal{C}$ [TE90] (avec $\lambda_j > 0$ pour tout j) :

$$\mathcal{C} = \left\{ \mathbf{v} = \left(\sum_j \lambda_j \mathbf{I}_j \right) \mathbf{M} \text{ avec } \sum_j \lambda_j < 1 \right\}$$

avec $\mathbf{M} = \text{diag}(c(e), e \in E)$.

Soit $\mathbf{c} = (c(e), e \in E)$ le vecteur de capacités des arêtes. Soit $H(e)$ le nombre maximum d'arêtes qui peuvent être actives si l'arête e n'est pas active et soit $H = \max(1, \max_{e \in E} H(e))$. Soit $\mathbf{b} = (E[B_s(e)], e \in E)$.

Nous pouvons alors donner une condition suffisante de stabilité :

Théorème 24 *Sous l'hypothèse 1 et si $E[B_s(e)B_s(e')]$ est bornée pour toute paire $e, e' \in E$ alors l'*`Algorithme Log` *stabilise le vecteur $\mathbf{b}$ si $(CH/(C+H-1))(\mathbf{b}+\varepsilon\,\mathbf{c}) \in \mathcal{C}$ pour un certain $\varepsilon > 0$.*

Preuve : Notre preuve est une adaptation de la preuve du Théorème de [BV09] (remplacer la définition de l'ensemble $\mathcal{L}'(s)$ dans [BV09] par $\mathcal{L}'(s) = \{e \in E : q_{sC+1}(e) \geq LCc(e)\}$) car l'`Algorithme Log` se comporte comme un ordonnancement hybride lorsque $\mathcal{L}'(s) = E$ pour tout $s \geq 0$. □

En guise d'exemple, considérons un chemin $G = (V, E)$ pour un modèle d'interférence à distance d. Dans ce cas, $H = 2(d+1)$ car une arête est en conflit avec au plus $2(d+1)$ autres arêtes et $C = d+2$ car $d+2$ couleurs sont nécessaires et suffisantes pour assigner des couleurs valides aux arêtes pour le modèle d'interférence à distance d. Supposons que pour chaque arête $e \in E$, $\{A_t(e), t \geq 0\}$ est une séquence i.i.d avec $\overline{A}(e) = E[A_t(e)]$ et que $E[A_t(e)A_t(e')]$ est bornée $\forall e, e' \in E$. En particulier, $E[B_s(e)] = 2\overline{A}(e)$. Le Théorème 24 dit que si le vecteur $\mathbf{a} = (\overline{A}(e), e \in E)$ est tel qu'il existe $\varepsilon > 0$ avec $(2(d+2)/3)((d+2)\mathbf{a}+\varepsilon\,\mathbf{c}) \in \mathcal{C}$ alors $\mathbf{a}$ stabilise l'`Algorithme Log`.

Conjecture 3 `Algorithme Log` *est stable si* $\forall e \in E$,

$$\sum_{e' \in \varepsilon(e) \cup \{e\}} \overline{A}(e') < \min_{e' \in \varepsilon(e) \cup \{e\}} c(e') \tag{4.2}$$

Cette conjecture pose la question de déterminer quelle est la région de capacité la plus grande parmi celles définies dans le Théorème 24 et par les conditions (4.2). Nous donnons une réponse partielle pour l'exemple décrit précédemment lorsque $c(e) = 1$ pour tout $e \in E$.

Supposons que $(2(d+2)/3)((d+2)\mathbf{a} + \varepsilon) \in \mathcal{C}$ pour $\varepsilon > 0$ de telle sorte que pour l'exemple précédent, l'`Algorithme Log` stabilise le vecteur $\mathbf{a} = (a(1), \ldots, a(|E|))$ avec $a(e) := \overline{A}(e)$.

De la définition de la région $\mathcal{C}$, nous savons qu'il existe des constantes $\lambda_i > 0$ avec $\sum_i \lambda_i < 1$ telles que

$$(2(d+2)/3)\left((d+2)a(e) + \varepsilon\right) = \sum_i \lambda_i \mathbf{I}_i(e), \quad e = 1, \ldots, |E|. \tag{4.3}$$

De (4.3) et comme $\varepsilon > 0$, nous déduisons que

$$a(e) < \frac{3}{2(d+2)^2} \sum_i \lambda_i \mathbf{I}_i(e), \quad e = 1, \ldots, |E|. \tag{4.4}$$

Nous montrons que le vecteur $\mathbf{a}$ réside strictement dans la région définie par les conditions de (4.2). Cela revient à montrer que $D(e) < 1$ pour $e = 1, \ldots, |E|$ avec

$$\begin{aligned} D(e) := a(e-1+d) + \cdots + a(e-1) + a(e) \\ + a(e+1) + \cdots + a(e+1+d) \end{aligned}$$

avec $a(e) = 0$ si $e \notin \{1, \ldots, |E|\}$ par convention. De (4.4) nous trouvons

$$\begin{aligned} D(e) < \frac{3}{2(d+2)^2} \sum_i \lambda_i (\mathbf{I}_i(e-1+d) + \cdots \\ + \mathbf{I}_i(e-1) + \mathbf{I}_i(e) + \mathbf{I}_i(e+1) + \cdots + \mathbf{I}_i(e+1+d)) \end{aligned}$$

pour $e = 1, \ldots, |E|$. Par définition du modèle d'interférence à distance d, nous observons que

$$\begin{aligned} \mathbf{I}_i(e-1+d) + \cdots + \mathbf{I}_i(e-1) + \mathbf{I}_i(e) \\ + \mathbf{I}_i(e+1) + \cdots + \mathbf{I}_i(e+1+d) \leq 2 \end{aligned}$$

pour $e = 1, \ldots, |E|$ car au plus 2 arêtes peuvent être simultanémenent actives parmi les $2(d+1)+1$ arêtes consécutives. Alors, $D(e) < 3/(d+2)^2 \leq 3/4$ pour $e = 1, \ldots, |E|$ et $\forall d \geq 0$. Cela montre que la région de stabilité donnée dans (4.2) est plus grande que celle donnée dans le Théorème 24.

Nous concluons cette section en observant que les conditions

$$\overline{A}(e) < c(e)/C, \quad e \in E. \tag{4.5}$$

assurent la stabilité d'`Algorithme Log` pour $K = 1$. En effet dans ce cas, les poids virtuels ne dépendent pas des valeurs des poids, ce qui implique qu'`Algorithme Log` devient un algorithme déterministe qui garantit que chaque arête e transmet au moins $c(e)$ messages toutes les C étapes consécutives.

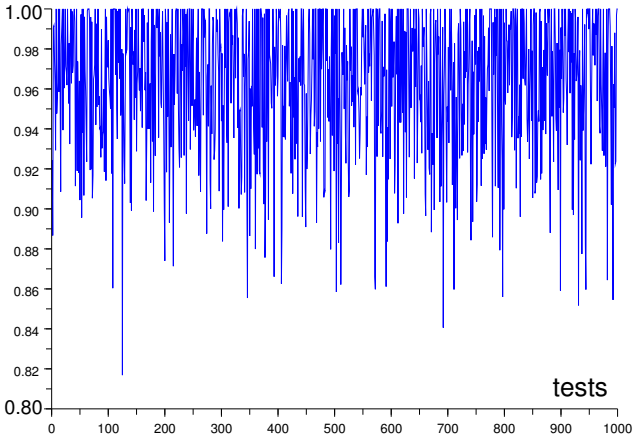


FIG. 4.7 – Rapport entre le couplage maximum et le couplage calculé par l'`Algorithme Log` pour un chemin de 50 arêtes avec le modèle d'interférence à distance $d = 0$ et les paramètres $C = 2$, $K = 50$ et $L = 50$.

4.4 Simulations

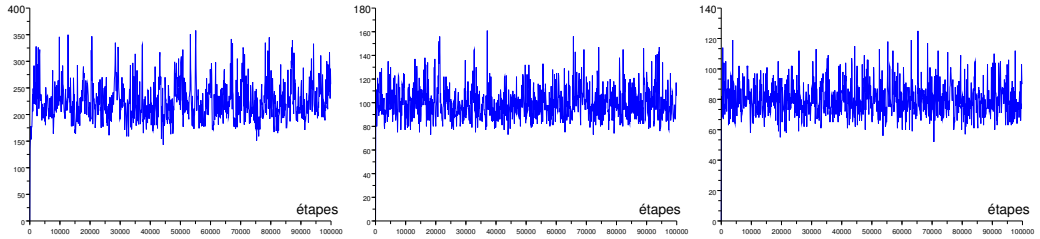
Nous étudions dans cette section les performances d'`Algorithme Log` via simulations. Plus précisément, nous calculons la somme des poids des arêtes actives à une étape donnée (section 4.4.1) et nous étudions l'évolution du poids de l'arête la plus chargée du graphe pour des milliers d'étapes (section 4.4.2). Par exemple nous comparons dans la section 4.4.2.2 le poids de l'arête la plus chargée pour l'`Algorithme Log` et l'`Algorithme des chemins augmentants`, proposé dans [BSS09].

4.4.1 Poids de l'ensemble des arêtes actives sur une étape

Nous considérons le modèle d'interférence à distance $d = 0$, un chemin $G = (V, E)$ composé de $|E| = 50$ arêtes, $K = 50$, $L = 50$ et $C = 2$. Nous attribuons un poids à chaque arête $e \in E$ comme suit : $P(q_t(e) = i) = 1/51$, $0 \leq i \leq 50$. Nous calculons le rapport entre le poids du couplage maximum et le poids du couplage calculé par l'`Algorithme Log` pour une étape donnée. La figure 4.7 décrit ce rapport pour 1000 tirages des poids. Nous observons qu'`Algorithme Log` calcule un couplage de poids au moins 4/5 du poids du couplage optimal.

4.4.2 Evolution des poids des arêtes sur un grand nombre d'étapes

Nous étudions dans cette section l'évolution des poids des arêtes pour un certain nombre d'étapes avec l'`Algorithme Log`. Les simulations ont été réalisées pour des chemins (section 4.4.2.1), des grilles (section 4.4.2.2) et des graphes aléatoires (section 4.4.2.3). Pour chacune des topologies, nous expliquons les paramètres choisis et décrivons l'évolution du poids maximum parmi toutes les arêtes sur plusieurs milliers d'étapes. De plus, dans la section 4.4.2.2, nous comparons l'`Algorithme Log` à



(a) $(n_1, n_2) = (16, 1)$ (b) $(n_1, n_2) = (12, 4)$ (c) $(n_1, n_2) = (8, 8)$

FIG. 4.8 – Evolution de l'arête la plus chargée avec l'`Algorithme Log` pour un chemin composée de 100 arêtes : modèle d'interférence à distance $d = 0$ et les paramètres $C = 4$, $K = 1000$ et $L = 1000$.

l'`Algorithme des chemins augmentants` pour une grille (poids maximum et poids moyen), en choisissant les paramètres de [BSS09].

4.4.2.1 Chemins

Comme décrit dans la section 4.3.3, nous pouvons associer la séquence d'entiers (couleurs) $1, 2, 1, 2, \ldots$ aux arêtes du chemin en considérant le modèle d'interférence à distance $d = 0$. En conséquence $C = 2$.

Nous analysons le comportement d'`Algorithme Log` pour un chemin $G = (V, E)$ composé de $|E| = 100$ arêtes pour 100000 étapes. Nous choisissons $K = 1000$ et $L = 1000$. De plus, pour chaque arête $e \in E$, la capacité est $c(e) = 18$ et les processus d'arrivées des messages sont définis comme suit : n_1 messages arrivent en moyenne sur e à chaque étape $t \geq 1$ pour chaque $e \in E$ avec couleur 1 ($\gamma(e) = 1$) et n_2 messages arrivent en moyenne pour les autres (chaque arête $e \in E$ telle que $\gamma(e) = 2$). La figure 4.8(a), la figure 4.8(b) et la figure 4.8(c) représentent l'arête de plus grand poids à chacune des étapes pour $(n_1, n_2) = (16, 1)$, $(n_1, n_2) = (12, 4)$ et $(n_1, n_2) = (8, 8)$, respectivement. Nous observons que le poids maximum est toujours plus petit que 400, 180 et 140, respectivement.

Nous évaluons ensuite l'impact de la valeur du paramètre K sur le poids de l'arête la plus chargée et sur le poids moyen. Le graphe considéré est un chemin de 100 arêtes avec le modèle d'interférence à distance $d = 0$. Pour chaque arête $e \in E$, $c(e) = 1$ et une moyenne de 0.75 (0.2, respectivement) messages arrive à l'étape $t \geq 1$ si $\gamma(e) = 1$ (si $\gamma(e) = 2$, respectivement). La figure 4.9 représente le poids maximum et le poids moyen après 50000 étapes pour différentes valeurs de K : $K = 1, 2, 4, 8, 16, 32, 64, 128$. Nous choisissons $L = K$. La figure illustre le compromis à réaliser entre la performance et le nombre de mini-slots de contrôle T_{log}. Une grande valeur de K donne des files d'attente peu chargées mais les gains les plus importants sont réalisés pour des petites valeurs de K (voir figure 4.9).

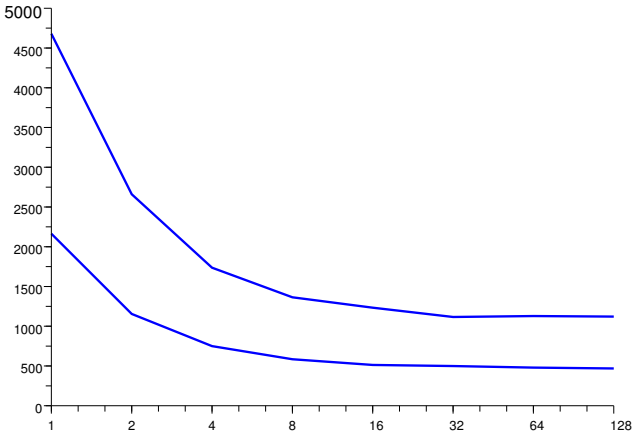
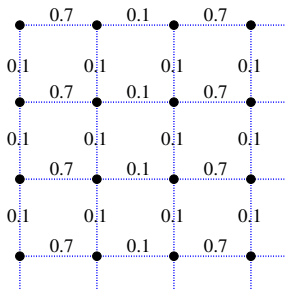


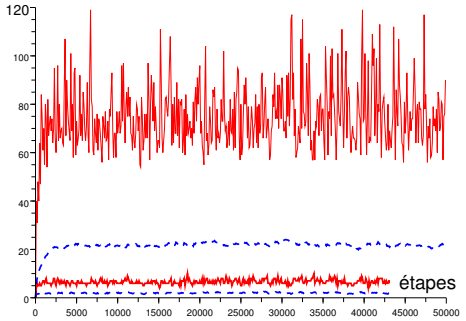
FIG. 4.9 – Soit un chemin de 100 arêtes après avec le modèle d'interférence à distance $d = 0$ et $C = 2$. Nous avons $c(e) = 1$ pour tout $e \in E$, 0.75 arrivées en moyenne pour les arêtes de couleur 1 et 0.2 arrivées en moyenne pour les arêtes de couleur 2. La figure représente le poids maximum parmi les arêtes après 50000 étapes en fonction de K : $K = 1, 2, 4, 8, 16, 32, 64, 128$ avec $L = K$.

4.4.2.2 Grilles

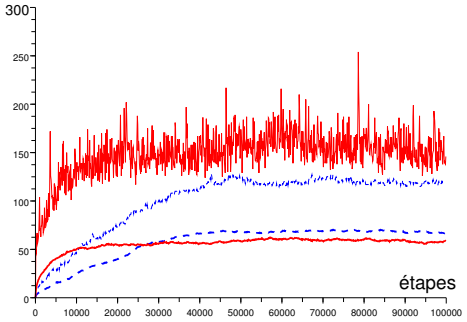
Nous comparons dans cette section l'`Algorithme Log` à l'`Algorithme des chemins augmentants`, choisissant les mêmes paramètres que dans [BSS09]. Plus précisément, les simulations ont été réalisées pour une grille $G = (V, E)$ composée de $|V| = 121$ sommets et de $|E| = 220$ arêtes. Pour chaque arête $e \in E$, la capacité est $c(e) = 1$. De plus, les valeurs mentionnées dans la figure 4.10(a) représentent les arrivées moyennes sur chacune des arêtes divisées par un paramètre $\lambda \leq 1$. Comme décrit dans la section 4.1.3, l'`Algorithme des chemins augmentants` nécessite le choix de deux principaux paramètres : k et p. Le paramètre k est lié à la taille des chemins augmentants et p représente la probabilité qu'un nœud du réseau soit choisi pour débuter la construction d'un chemin augmentant. Dans [BSS09], les simulations ont été réalisées pour $\{k = 2, p = 0.2\}$, $\{k = 3, p = 0.2\}$ et $\{k = 3, p = 0.1\}$. Rappelons que l'`Algorithme des chemins augmentants` est valide seulement pour le modèle d'interférence à distance $d = 0$ et donc nous utilisons ce modèle pour nos comparaisons. Décrivons maintenant les paramètres choisis pour l'`Algorithme Log`. De la section 4.3.3, nous savons que la valeur minimum de C respectant les contraintes décrites précédemment dans la section 4.2 est $\chi'(G) = 4$. Voir la figure 4.5(b) pour un exemple d'assignations valides de $\gamma(e)$. De plus, nous choisissons $K = 1000 = L + 1$. Nous comparons dans la figure 4.10 la valeur de l'arête la plus chargée et la valeur moyenne des arêtes pour chaque étape $t \leq 50000$ avec l'`Algorithme Log` et l'`Algorithme des chemins augmentants` et pour les paramètres $\{k = 2, p = 0.2\}$ décrit dans [BSS09]. Les résultats de simulations pour les deux autres jeux de paramètres sont analogues. La figure 4.10(b), la figure 4.10(c) et la figure 4.10(d) représentent l'arête la plus chargée et la moyenne des poids pour



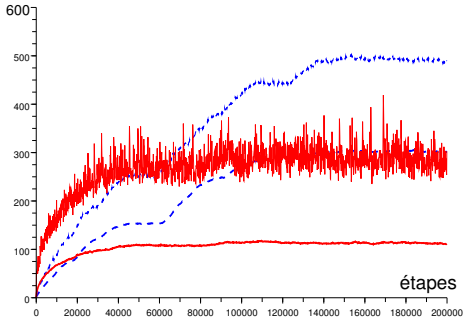
(a) Grille.



(b) $\lambda = 0.90$



(c) $\lambda = 0.95$



(d) $\lambda = 0.97$

FIG. 4.10 – Soit la grille composée de 121 sommets et de 200 arêtes avec le modèle d'interférence à distance $d = 0$ et $C = 4$. La figure représente l'évolution de l'arête la plus chargée et de la valeur moyenne des poids avec l'`Algorithme des chemins augmentants` ($k = 2$ et $p = 0.2$) et avec l'`Algorithme Log`.

$\lambda = 0.90, 0.95, 0.97$, respectivement. La courbe rouge pleine représente l'`Algorithme des chemins augmentants` et la courbe bleue en pointillés représente l'`Algorithme Log`. Rappelons que l'`Algorithme des chemins augmentants` est propre au modèle d'interférence à distance $d = 0$ alors que l'`Algorithme Log` semble stabiliser le système pour d'autres valeurs de d.

En effet, la figure 4.11 montre une simulation pour la même grille avec le modèle d'interférence à distance $d = 1$ avec les paramètres $K = L = 1000$ et des capacités et des processus d'arrivées respectant les conditions de la Conjecture 3. Nous observons que l'`Algorithme Log` semble stabiliser le système.

4.4.2.3 Graphes aléatoires

Nous évaluons dans cette section les performances d'`Algorithme Log` pour un réseau aléatoire. Décrivons tout d'abord la manière dont nous avons construit nos *graphes aléatoires* avant d'analyser l'évolution de la file d'attente la plus chargée.

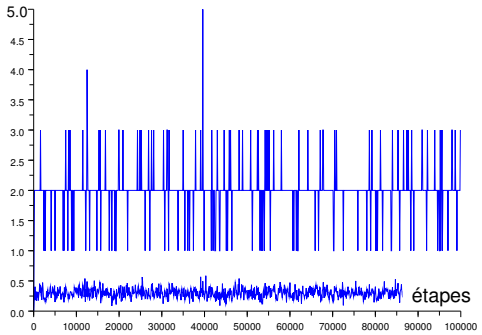


FIG. 4.11 – Evolution de l'arête la plus chargée et de la valeur moyenne des poids pour une grille composée de 121 sommets et de 200 arêtes avec le modèle d'interférence à distance $d = 1$ avec les paramètres $K = L = 1000$ et des capacités et des processus d'arrivées respectant les conditions de la Conjecture 3.

Etant donnée une grille $G' = (V, E')$, nous supprimons une arête $e \in E'$ avec une certaine probabilité constante q_1, avant d'ajouter d'autres arêtes avec une certaine probabilité constante q_2 entre deux sommets à une certaine distance euclidienne. Nous obtenons un graphe $G = (V, E)$. La figure 4.6 montre un exemple de graphe aléatoire avec $|V| = 33$ sommets et $|E| = 56$ arêtes.

Dans nos simulations, nous vérifions empiriquement le critère de stabilité décrit par l'équation 4.2 de la Conjecture 3 avec $c(e) = 20, \forall e \in E$.

Les processus d'arrivées des messages sur les arêtes du graphe sont définis comme suit : nous associons premièrement à chacune des arêtes $e \in E$ une moyenne d'arrivées de 20 messages par étape $t \geq 1$ (loi uniforme entre 0 et 40). Afin de respecter les conditions précédentes, nous réduisons le nombre moyen d'arrivées de 1 pour une arête si nécessaire, jusqu'à obtenir la condition désirée. Dans la figure 4.12, nous appliquons l'`Algorithme Log` pour un graphe aléatoire $G = (V, E)$ composé de $|V| = 200$ sommets et de $|E| = 424$ arêtes. Le nombre d'étapes est 100000 avec $K = 10000$, $L = K$, $C = 20$ et le modèle d'interférence à distance $d = 0$. La figure 4.12 représente le poids de l'arête de poids maximum pour chacune des étapes. Nous observons qu'`Algorithme Log` semble stabiliser le système des files d'attente associées aux arêtes.

4.5 Conclusion et perspectives

Nous proposons un algorithme distribué ou local pour le problème d'ordonnancement des liens dans les réseaux sans-fil avec les propriétés suivantes : phase de contrôle constante en la taille du réseau et en les tailles des files d'attente associées aux liens, valide pour tout modèle d'interférence binaire, ensemble maximal de liens actifs à chaque étape. Nous montrons une condition suffisante pour obtenir la stabilité du système de files d'attente même si nous en conjecturons une meilleure (Conjecture 3).

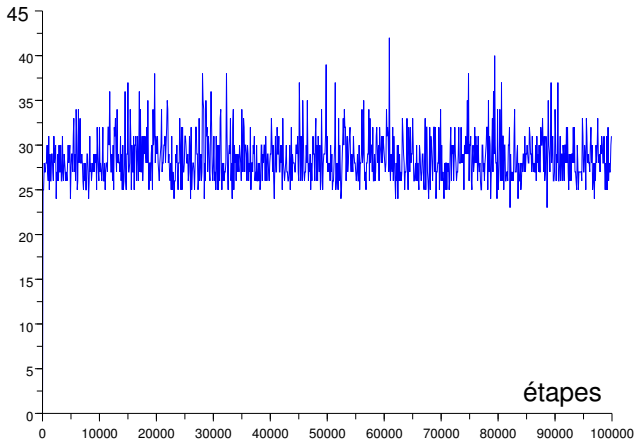


FIG. 4.12 – Soit le graphe généré aléatoirement composé de 200 sommets et 424 arêtes avec le modèle d'interférence à distance $d = 0$ et $C = 20$. Nous avons $c(e) = 20$ pour tout $e \in E$. Nous assignons initialement 20 arrivées en moyenne sur les arêtes (uniforme) et nous réduisons cette moyenne de 1 jusqu'à obtenir les conditions de la Conjecture 3. La figure représente le poids maximum parmi les arêtes pour 100000 étapes avec les paramètres $K = L = 10000$.

Nous pouvons prendre en compte le trafic multi-hop dans l'`Algorithme Log` lorsque le routage est fixé. L'idée est de changer le vecteur q_t des poids et le vecteur q'_t des poids virtuels (section 4.2.1) en un vecteur des poids multi-hop q_t^m et en un vecteur des poids virtuels multi-hop $q_t'^m$, respectivement. Soit $G = (V, E)$ le graphe représentant le réseau. Plus précisément, $\forall e \in E$, $q_t^m(e) = \sum p_t^i(e) h_t^i(e)$ avec $p_t^i(e)$ le nombre de messages indicés i dans la file d'attente e à l'étape t et avec $h_t^i(e)$ le nombre de sauts restants de ces messages. A partir de $q_t^m(e)$, nous calculons $q_t'^m(e)$ pour $e \in E$ de la même manière que pour l'`Algorithme Log` (section 4.2.1). Nous supposons ici que le routage est défini préalablement et ne change pas. Remarquons qu'en fonction de la politique de services souhaitée, les calculs peuvent différer.

En travail futur, il est nécessaire d'étudier les performances d'`Algorithme Log` pour d'autres jeux de paramètres, notamment pour des modèles d'interférence variés. Il est également intéressant de poursuivre nos études sur la stabilité des files d'attente avec l'`Algorithme Log` dans le but de prouver, entre autres, la Conjecture 3. Toutes nos fonctions de Lyapunov candidates n'ont pas permis d'obtenir le résultat désiré. Enfin, nous continuerons ce travail en prenant en compte un trafic multi-hop sans nécessairement avoir un routage prédéfini.

Chapitre 5

Placement de données dans les réseaux pair-à-pair

Sommaire

5.1 Introduction

Nous présentons deux travaux que nous avons réalisés concernant les systèmes de stockage de données dans les réseaux pair-à-pair. Nous étudions premièrement l'influence de trois politiques de placement sur le temps moyen avant la première perte de données dans les systèmes avec redondance. Ensuite, nous étudions les placements dans les systèmes avec réplication dans le but de minimiser la variance du nombre de données indisponibles (l'espérance ne variant pas selon le placement).

Ce travail a été réalisé avec Jean-Claude Bermond, Stéphane Caron, Frédéric Giroire, Alain Jean-Marie, Julian Monteiro, Stéphane Pérennes et Joseph Yu [CGM+10b, CGM+10c, CGM+10a, BJMMY11].

Durée de vie des données pour trois politiques de placement.

Le principe des systèmes de stockage de données pair-à-pair est de distribuer les données parmi les pairs dans le but d'assurer la fiabilité et une bonne tolérance aux pannes.

Une donnée est divisée en blocs. Chaque bloc est divisé en s fragments. Le codage produit $s + r$ fragments pouvant ainsi tolérer r pannes. En d'autres termes, le bloc original peut être reconstruit à partir de n'importe quels s fragments parmi les $s+r$. Dans [LCZ05, GMP09], les auteurs montrent que la manière de placer les fragments sur les pairs a une grande influence sur les performancecs du système. Nous étudions le temps moyen avant que le système perde une donnée (*Mean Time to Data Loss*, MTTDL) pour trois politiques de placement différentes.

Dans la première (*global policy*), les fragments sont envoyés sur des pairs choisis uniformément au hasard parmi les N pairs du réseau. Dans la deuxième (*buddy policy*), les pairs sont divisés en C clusters de taille $s + r$ chacun. Les fragments sont

ensuite envoyés sur un cluster choisi uniformément au hasard parmi les C clusters. Dans cette politique, tous les pairs d'un cluster stockent le même ensemble de blocs. Enfin, dans la dernière politique de placement (*chain policy*), le réseau est vu comme un anneau dirigé de N pairs et les fragments sont envoyés sur $s+r$ pairs consécutifs uniformément au hasard. L'avantage de la politique globale est une meilleure répartition de la charge parmi les pairs, permettant notamment une reconstruction rapide [GMP09]. Cependant les deux politiques locales offrent certains avantages pratiques notamment concernant le stockage de méta-informations [CDH+06, DLS+04].

Le temps est divisé en étapes toutes de même taille. Comme dans la plupart des études, nous considérons qu'un pair est indisponible ou tombe en panne avec une probabilité constante α. Nous supposons l'indépendance des pannes. La reconstruction des fragments d'un bloc démarre dès qu'un des ses fragments est perdu (*eager reconstruction strategy*). De plus, nous supposons que tous les fragments sont reconstruits en une étape. Une fois cette reconstruction effectuée, les fragments sont redistribués sur différents pairs, selon la politique de placement utilisée. Ainsi, un bloc est perdu lorsque $r+1$ de ses fragments sont perdus durant une même étape. La donnée contenant ce bloc est alors considérée comme perdue. Dans [CGM+10b], nous traitons le problème pour différents mécanismes de reconstruction.

Il est possible d'observer que le nombre moyen de données perdues lors d'une étape est identique pour les trois politiques de placement décrites précédemment. En revanche, comme décrit dans [GMP09], la durée moyenne avant la première perte de données varie beaucoup selon la politique utilisée. En effet, pour la politique globale, la perte de données arrive régulièrement alors que pour la politique utilisant les clusters, cette perte est peu fréquente. Mais lorsqu'il y a une perte de données pour cette politique, le nombre de pertes est très important (tous les blocs du cluster concerné sont perdus). Pour la politique de placement selon un anneau logique (*chain policy*), la durée moyenne avant la première perte a un comportement entre les deux. Nous proposons des méthodes pour calculer cette durée moyenne de manière exacte ou de manière approchée pour les trois politiques de placement.

Nous présentons nos résultats dans la section 5.2.

Configurations bien équilibrées pour le placement de données.

Dans le deuxième article, nous considérons le problème de placement de données motivé par la réplication de données dans un système de vidéo à la demande (voir [BBJMR09a, BBJMR09b, JMRBB09]). Nous avons un ensemble de n serveurs et de b fichiers (données, documents, fragments). Chaque fichier est répliqué sur exactement k serveurs. L'ensemble des serveurs contenant un fichier donné a donc taille k et sera appelé (comme dans la théorie des configurations) un bloc B. Un placement est donc constitué par une famille $\mathcal{F}$ de b blocs.

Un serveur est supposé disponible avec une certaine probabilité δ et donc indisponible (en panne) avec la probabilité $1-\delta$. Un fichier est disponible si au moins un serveur le contenant est disponible ou de manière équivalente le fichier est indisponible (perdu) si tous les serveurs le contenant sont indisponibles. Dans [JMRBB09], les auteurs étudient la variable aléatoire Λ du nombre de fichiers disponibles. Ils

prouvent que l'espérance $E(\Lambda)$ est indépendante du placement mais pas sa variance. Ils montrent que minimiser la variance revient à minimiser le polynôme $P(\mathcal{F}, x) = \sum_{j=0}^{k} v_j x^j$, où v_j représente le nombre de couples de blocs qui s'intersectent en exactement j éléments et où $x = \frac{1}{1-\delta}$ (donc $x \geq 1$). Le problème que nous considérons consiste, pour un nombre n de serveurs, b de fichiers et un paramètre k de réplication, à trouver un placement optimal pour la variable x, c'est-à-dire une famille $\mathcal{F}$ qui minimise le polynôme $P(\mathcal{F}, x)$.

Les auteurs de [JMRBB09] conjecturent qu'il existe un placement optimal quel que soit la valeur de $x \geq 1$ (c'est-à-dire une famille $\mathcal{F}^*$ telle que $P(\mathcal{F}^*, x) \leq P(\mathcal{F}, x)$ pour tout $\mathcal{F}$ et tout $x \geq 1$).

Nous montrons dans cet article que le polynôme $P(\mathcal{F}, x)$ peut s'écrire sous la forme $\Sigma_{j=1}^{k} \Sigma_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j} (x-1)^j - bx^k + b^2$ où $\lambda_{x_1,\ldots,x_j}$ représente le nombre de blocs contenant $\{x_1, \ldots, x_j\}$. Nous disons qu'une famille $\mathcal{F}$ est j-équilibrée si les $\lambda_{x_1,\ldots,x_j}$ sont tous égaux ou quasi égaux, c'est-à-dire $|\lambda_{x_1,\ldots,x_j} - \lambda_{y_1,\ldots,y_j}| \leq 1$ pour tout $\{x_1, \ldots, x_j\}$ et $\{y_1, \ldots, y_j\}$. La famille sera dite "bien équilibrée" si elle est j-équilibrée pour tout j, $1 \leq j \leq k$. Une famille j-équilibrée minimise le coefficient $\Sigma_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j}$ et donc une famille bien équilibrée correspond à un placement optimal et la conjecture est vérifiée pour les valeurs de n, b, k pour lesquelles il existe une famille bien équilibrée. Dans le cas $k = 2$, il est facile de construire une famille bien équilibrée en utilisant des couplages du graphe complet. Pour $k \geq 3$, le problème se révèle difficile. En effet, un cas particulier du problème correspond à l'existence de systèmes de Steiner. Ainsi pour $k = 3$, un système de triplets de Steiner ([CJ06]) $(n, 3, 1)$ est défini comme une famille de blocs telle que chaque paire d'éléments appartient à exactement 1 bloc. De tels systèmes existent si et seulement si $n \equiv 1$ ou $3 \pmod 6$. Dans ce cas, $b = \frac{n(n-1)}{6}$ et chaque sommet appartient à exactement $\frac{n(n-1)}{2}$ blocs.

Dans l'article, nous donnons des méthodes pour construire pour $k = 3$, des familles bien équilibrées. Nous utilisons l'existence pour $n = 6t + 3$ de *Kirkman resolvable triple systems* où chaque $(n, 3, 1)$ configuration peut elle même être décomposée en $3t + 1$ classes parallèles (une classe parallèle étant formée de $2t + 1$ triplets formant une partition des n éléments).

Nous présentons nos résultats dans la section 5.3.

5.2 Durée de vie des données pour trois politiques de placement

Nous proposons des méthodes exactes et approchées pour calculer le temps moyen avant la première perte de données selon les trois politiques de placement présentées précédemment. Nous présentons nos résultats parus dans l'article [CGM+10a].

Data Life Time for Different Placement Policies in P2P Storage Systems *

S. Caron[1], F. Giroire[2], D. Mazauric[2], J. Monteiro[2], and S. Pérennes[2]

[1] ENS Paris
[2] MASCOTTE joint team, INRIA, I3S (CNRS, Univ. of Nice)

Abstract. Peer-to-peer systems are foreseen as an efficient solution to achieve reliable data storage at low cost. To deal with common P2P problems such as peer failures or churn, such systems encode the user data into redundant fragments and distribute them among peers. The way they distribute it, known as *placement policy*, has a significant impact on their behavior and reliability. In this paper, we study the impact of different placement policies on the data life time. More precisely, we describe methods to compute and approximate the mean time before the system loses data (*Mean Time to Data Loss*). We compare this metric for three placement policies: two of them *local*, in which the data is stored in logical peer neighborhoods, and one of them *global* in which fragments are parted uniformly at random among the different peers.

1 Introduction and System Description

The key concept of Peer-to-Peer storage systems is to distribute redundant data among peers to achieve high reliability and fault tolerance at low cost. The addition of redundant data could be done by *Erasure Codes* [14], such as Reed Solomon, as used by some RAID schemes. When using Erasure Codes, the original user data (e.g. files, raw data, etc.) is cut into *blocks* that are in turn divided into s initial *fragments*. The encoding scheme produces $s + r$ fragments that can tolerate r failures. In other words, the original block can be recovered from any s of the $s + r$ encoded fragments. In a P2P storage system, these fragments are then placed on $s + r$ different peers of the network according to a placement policy, which is the main subject of this paper. In [8] we studied placement policies by simulations, and we presented the amount of resource (bandwidth and storage space) required to maintain redundancy and to ensure a given level of reliability. In this paper, we present an analytical method to compute the metric Mean Time to Data Loss (MTTDL) for three different placement policies. The remainder of this paper is organized as follows: first we briefly present the characteristics of the studied P2P storage systems, followed by the related work. In Section 2, we describe the studied placement policies. Then, in Sections 3, 4, 5, we describe the analytical methods to compute exact values and approximations of the MTTDL for the three policies. We conclude in Section 6.

Peer Failures. It is assumed that the peers stay connected almost all the time into the system. Indeed, in our model a peer failure represents a disk crash or a peer that definitively leaves the system. In both cases, it is assumed that all the data on the peer's disk are lost. Following most works on P2P storage systems, peers get faulty

* This work was partially funded by the ANR project SPREADS and région PACA.

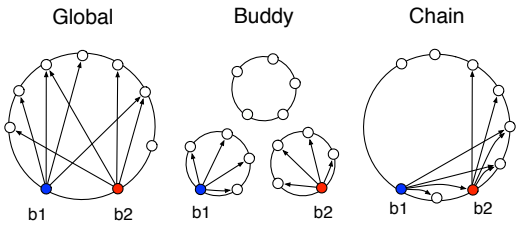


Fig. 1. Placement of two blocks $b1$ and $b2$ in the system using the different policies.

independently according to a memoryless process. For a given peer, the probability to fail at a given time step is α.

Reconstruction Strategy. To ensure a durable long-term storage despite disk failures, the system needs to continuously monitor the number of fragments of each block and maintain a minimum number of redundancy fragments available in the network. In this work, we study the case where the reconstruction starts as soon as one of its fragments is lost, namely *eager* reconstruction strategy. In addition, the blocks are reconstructed in one time step, i.e., there is enough bandwidth to process the reconstruction quickly. After the reconstruction, the regenerated missing fragments are spread among different peers. Hence, after each time step, the system is fully reconstructed. We also studied systems with other reconstruction processes in [2], but we do not discuss them here due to lack of space.

Related Work

The majority of existing or proposed systems, e.g., CFS, Farsite [6], PAST, TotalRecall [1], use a local placement policy. For example, in PAST [13], the authors use the Pastry DHT to store replicas of data into logical neighbors. In the opposite way, some systems use a Global policy, as OceanStore [11] or GFS [7]. GFS spreads chunks of data on any server of the system using a pseudo-random placement. Chun et al. in [3] and Ktari et al. in [10] discuss the impact of data placement. The later do a practical study of a large number of placement policies for a system with high churn. They exhibit differences of performance in terms of delay, control overhead, success rate, and overlay route length. In the work closer to ours [12], the authors study the impact of data placement on the Mean Time to Data Loss (MTTDL) metric. All these studies consider the case of systems using replication. In this paper, we address the more complex case of Erasure Codes which are usually more efficient for the same storage overhead [14].

2 Placement Policies

It has been shown that fragment placement has a strong impact on the system performance [8,12]. We study here three different strategies to place the $s + r$ fragments of a block, as explained in the following and depicted in Figure 1:

- *Global Policy:* fragments are sent to peers chosen uniformly at random among all the N peers.
- *Buddy Policy:* peers are grouped into C independent clusters of size exactly $s + r$ each. The fragments are then sent to a cluster chosen uniformly at random among the

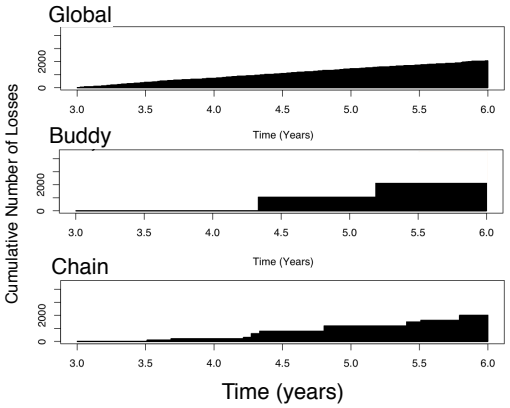


Fig. 2. Illustrative example of the cumulative number of block losses for a period of three years. The number of losses is the same among policies, but its distribution over time is different.

clusters. In this situation, all peers of a cluster store fragments of the same set of blocks. It could be seen as a collection of local RAID like storage.

• *Chain Policy:* the network is seen as a directed ring of N peers and the fragments are sent to $s+r$ consecutive peers choosen uniformly at random. This policy corresponds to what is done in most distributed systems implementing a DHT.

The use of the Global policy allows the system to distribute more uniformly the load among peers, leading to a faster reconstruction and a smoother operation of the system [8]. However, the use of Buddy and Chain, namely *local* strategies, brings practical advantages [4,3]. For example, the management traffic and the amount of meta-information to be stored by the peers are kept low.

Data Loss Rate. A data loss occurs when at least one block is lost. A block is considered lost if it loses at least $r+1$ fragments during one time step, otherwise, recall that all the $s+r$ fragments are fully reconstructed at next time step. The data loss rate for a given block comes straightforward. This loss rate does not depend on the placement policy (as soon as it is assured that all fragments are stored on different peers). Hence, we have the same expected number of lost blocks for the three placement policies.

Mean Time to Data Loss (MTTDL). However, as stated in [8], the measure of the time to the first occurrence of a data loss shows that the three policies have very distinct behaviors. It is shown by simulations that the average quantity of data loss per year is the same, but the distribution across time of these losses is very different (see Figure 2). In the Global policy the losses occurs regularly. Conversely, they occur very rarely for the Buddy placement, but, when they occur, they affect a large batch of data. Basically, all the blocks of a Buddy cluster are lost at the same time. The behavior of the Chain policy is somewhere in the middle of both. In the next section we propose analytical methods to compute these variations through the metric MTTDL.

3 Buddy Placement Policy

In the next three sections (Section 3, 4 and 5), we present methods to compute exact values and approximations of the MTTDL for the three placement policies. For each policy, we calculate the probabilty $\mathbb{P}_{policy}$ to lose data at any given time step. Then, we deduce $MTTDL_{policy} = 1/\mathbb{P}_{policy}$.

In the Buddy placement policy, the N peers are divided into C clusters of size $s+r$ each. In this strategy, the calculation of the MTTDL_{buddy} is straightforward. Given a cluster, the probability to have a block loss is the probability that the cluster loses at least $r+1$ peers (i.e., fragments), is given by

$$\mathbb{P}_{cluster} = \sum_{j=r+1}^{s+r} \binom{s+r}{j} \alpha^j (1-\alpha)^{s+r-j}. \tag{1}$$

In fact, when that happens all the data stored on that cluster is lost. Remember that α is the probability of a given peer to fail at one time step. Since all the C clusters are independent, the probability to have a data loss is given by $\mathbb{P}_{buddy} = 1-(1-\mathbb{P}_{cluster})^C$.

If the average number of cluster failures per time step $C{\cdot}\mathbb{P}_{cluster} \ll 1$, as expected in a real system (i.e., the probabilty of simultaneous cluster failures is small), then we have $\mathbb{P}_{buddy} \approx C{\cdot}\mathbb{P}_{cluster}$, and so $\text{MTTDL}_{buddy} \approx 1/(C \cdot \mathbb{P}_{cluster})$.

If $(s+r)\alpha \ll 1$, we can approximate even more. In other words, this assumption means that the probability of a peer failure α is small. Since the ratio between two consecutive terms in sum of Equation (1) is $\leq (s+r)\alpha$, we can bound its tail by a geometric series and see that it is of $O((s+r)\alpha)$. We obtain $\mathbb{P}_{cluster} \approx \binom{s+r}{r+1}\alpha^{r+1}$. Then we have

$$\text{MTTDL}_{buddy} \approx \frac{1}{\frac{N}{s+r} \cdot \binom{s+r}{r+1}\alpha^{r+1}}. \tag{2}$$

4 Global Placement Policy

In the Global policy, block's fragments are parted between $s+r$ peers chosen uniformly at random. First, we present the exact calculation of the MTTDL_{global}. We then present approximated formulas that give an intuition of the system behavior.

4.1 MTTDL calculation

First, we consider i failures happening during one time step. Let F denote the set of the placement classes (i.e., groups of $s+r$ peers) that hold at least $r+1$ of these i failures; we have:

$$\#F = \sum_{j=r+1}^{i} \binom{i}{j}\binom{N-i}{s+r-j} \tag{3}$$

Then, suppose we insert a new block in the system: his $s+r$ fragments are dispatched randomly in one of the $\binom{N}{s+r}$ placement classes with uniform probability. Thus, the probability $\mathbb{P}_{block}(i)$ for the chosen class to be in F is:

$$\mathbb{P}_{block}(i) := \mathbf{P}\left[\text{placement in } F\right] = \frac{\sum_{j=r+1}^{i}\binom{i}{j}\binom{N-i}{s+r-j}}{\binom{N}{s+r}}$$

As block insertions are *independent*, if we consider our B blocks one after the other, the probability that none of them falls in F is $(1-\mathbb{P}_{block}(i))^B$. We then come back to the global probability to lose data considering different failure scenarii:

$$\begin{aligned}\mathbb{P}_{global} := \mathbf{P}\left[\text{lose data}\right] &= \mathbf{P}\left[\bigcup_{\{i \text{ failures}\}}\left[\text{failure kills a block}\right]\right] \\ &= \sum_{i=r+1}^{N}\binom{N}{i}\alpha^i(1-\alpha)^{N-i}\mathbf{P}\left[i \text{ failures kill a block}\right]\end{aligned}$$

Which gives us the MTTDL of the system using the global policy:

$$\text{MTTDL}_{global}^{-1} \approx \sum_{i=r+1}^{N}\binom{N}{i}\alpha^i(1-\alpha)^{N-i}\left[1-\left(1-\frac{\sum_{j=r+1}^{i}\binom{i}{j}\binom{N-i}{s+r-j}}{\binom{N}{s+r}}\right)^B\right] \tag{4}$$

4.2 MTTDL approximation

We provide here approximations for systems with low peer failure rates: *backup systems*. One example is *Brick storage systems* [12]. Each peer is a "brick" dedicated to data storage, that is, a stripped down computer with the fewest possible components: CPU, motherboard, hard drive and network card. In these backup systems, as we want a very high data life time, we have either $\alpha N \ll 1$ or $\alpha N \sim 1$, i.e., we have a not too high mean number of peer failures per time step.

Computations of this complicated sum suggests that only its first terms matter, and especially the very first term when $\alpha N \ll 1$. We can formalize this: let us consider three "zones" for $i \in [\![r+1, N]\!]$: (I) $i \sim s+r$, (II) $s+r \ll i \ll N$ and (III) $i \sim N$. We introduce the following notations:

$$\begin{aligned} A_i &= \textstyle\sum_{j=r+1}^{s+r}\binom{i}{j}\binom{N-i}{s+r-j} \;;\; C_i = 1-\frac{A_i}{\binom{N}{s+r}} \\ \Gamma_i &= 1-C_i^B \qquad\qquad\quad ;\; \Delta_i = \textstyle\binom{N}{i}\alpha^i(1-\alpha)^{N-i}\Gamma_i \end{aligned}$$

Where A_i is nothing but $\#F$ in case i failures happen. In fact, and for the sake of curiosity, we can compute it easily with the following relation.

Lemma 1. *For $i \geq r+1$, $A_{i+1} = A_i + \binom{i}{r}\binom{N-(i+1)}{s-1}$.*

Proof. F is the set of placement classes with at least $r+1$ of them falling into a given "failure" set of size i. Let us see what happens when we increment the size of this failure set. We denote by S_i the initial failure set of F and $S_{i+1} = S_i \cup \{x\}$. A placement class falls in S_{i+1} iff it has at least $r+1$ peers in it, which is equivalent to either (a) having

more that $r+1$ peers in S_i or (b) containing x and *exactly* r peers in S_i (cases where there are more than $r+1$ peers in S_{i+1}, including x, are already counted in (a)). From this we conclude that: $A_{i+1} = A_i + \binom{i}{r}\binom{N-(i+1)}{s-1}$.

The ratio between two consecutive terms of sum (4) is:

$$\rho := \frac{\Delta_{i+1}}{\Delta_i} \;=\; \frac{\alpha}{1-\alpha}\frac{N-i+1}{i+1}\frac{\Gamma_{i+1}}{\Gamma_i} \;\approx\; \alpha N \cdot \frac{\Gamma_{i+1}}{i\Gamma_i} \tag{5}$$

In zones (II) and (III), we can show this ratio is low enough so we can bound the tail of our sum by a geometric series of common ration $\rho \ll 1$.

Lemma 2. *In zone (I), under the assumption* $\frac{N}{(s+r)^2} \gg 1$*,*

$$\Delta_i \approx B\binom{s+r}{r+1}(\alpha N)^{i-(r+1)}\alpha^{r+1}(1-\alpha)^{N-i} \tag{6}$$

Proof. When $i \sim s+r$, we usually (read: in practice) have $A \Big/ \binom{N}{s+r} \ll 1$. Under our (strong) assumption, which is also verified in practice, we indeed have the simple bound $A \Big/ \binom{N}{s+r} \leq \left(\frac{(s+r)^2}{N}\right)^{r+1} \frac{s}{(r+1)!} \ll \frac{1}{B}$. Thus, Γ_i is almost proportional to C_i in zone (I), which implies $\Delta_i \approx B\alpha^i(1-\alpha)^{N-i}A\binom{N}{i} \Big/ \binom{N}{s+r}$. But simple combinatorics show that $A\binom{N}{i} = \sum_{j=r+1}^{s+r}\binom{s+r}{j}\binom{N-(s+r)}{i-j}\binom{N}{s+r}$, leading us to equation (6).

Lemma 3. *In zone (II),* $\rho \approx \frac{\alpha N}{i}$.

Proof. When $s+r \ll i \ll N$, we have

$$\begin{aligned} A_i &\approx \sum_{j=r+1}^{s+r} \frac{i^j}{j!}\frac{(N-i)^{s+r-j}}{(s+r-j)!} \\ C_i &\approx \left(1-\frac{i}{N}\right)^{s+r}\sum_{j=0}^{r}\binom{s+r}{j}\left(\frac{i}{N-i}\right)^j \\ &\approx \sum_{j=0}^{r}\binom{s+r}{j}\left(\frac{i}{N}\right)^j \sum_{l=0}^{s+r-j}(-1)^l\left(\frac{i}{N}\right)^l \end{aligned}$$

Taylor expansion to second order in $\frac{i}{N}$ leads us to $\Gamma_i \approx B\left[2(s+r)-3\right]\left(\frac{i}{N}\right)^2$. Hence we see that $\frac{\Gamma_{i+1}}{\Gamma_i} \approx \left(1+\frac{1}{i}\right)^2 \approx 1$, equation (5) leading us to $\rho \approx \alpha N/i$.

Lemma 4. *In zone (III),* $\rho \leq \frac{\alpha N}{i}$.

Proof. Let $\epsilon_i = 1 - \frac{i}{N}$: when $i \sim N$, we have $C_i \approx \sum_{j=0}^{r}\left(\frac{i}{N}\right)^j \epsilon_i^{s+r-j}\binom{s+r}{j} \approx \epsilon_i^s\binom{s+r}{r}$. Hence, $C_{i+1} - C_i \approx \frac{1}{N^s}\left(\epsilon_{i+1}^{s-1} + \cdots + \epsilon_i^{s-1}\right)\binom{s+r}{r} \leq \frac{1}{N^s}s\epsilon_i^{s-1}\binom{s+r}{r} \ll 1$. Then, Taylor expansion of the convex function $f(x) = 1 - x^B$ leads us to ($f'' < 0$):

$$\begin{aligned} \Gamma_{i+1} - \Gamma_i &\leq (C_{i+1} - C_i)f'(C_i) \\ &\leq \frac{1}{N^s}s\epsilon_i^{s-1}\binom{s+r}{r}BC_i^{B-1} \\ \frac{\Gamma_{i+1}}{\Gamma_i} &\leq 1 + \frac{B\epsilon_i^{s-1}s\binom{s+r}{r}}{N^s}\frac{C_i^{B-1}}{1-C_i^B} \end{aligned}$$

Since in practice we have $B \ll N^s$, this upper bound is close to 1 and we conclude – as usual – with equation (5) giving $\rho \leq \alpha N/i$.

Lemmas 3 and 4 tell us that, when $i \gg s + r$, our big sum is bounded by a geometric series of common ratio $\leq \frac{\alpha N}{i} \ll 1$, so only the terms before zones (II) and (III) numerically matter.

Lemma 2 can provide us with a stronger result. Equation (6) leads to $\rho \approx \alpha N$ in zone (I). Hence, if we also have $\alpha N \ll 1$, that is, mean number of failures per time step is really low (or, equivalently, time step is short enough), then only the first term of the sum matters. If we simplify it further, we find:

$$\text{MTTDL}_{global} \approx \frac{1}{B\binom{s+r}{r+1}\alpha^{r+1}} \tag{7}$$

5 Chain Placement Policy

For the Chain policy, the computation of MTTDL_{chain} is more difficult than the two previous ones, mainly because the chains are not independent of each other. From the definition of the Chain policy, a data loss occurs only when $r + 1$ (or more) peer failures are located at $s + r$ consecutive peers.

We present in this paper two approaches to compute or approximate the MTTDL for the Chain policy. We first describe computations using Markov chains techniques, and we then describe an analytical approximation value assuming that α is small enough.

5.1 Markov Chain Approach

The idea is to survey the N sequences $S_1, S_2, \ldots, S_N$ of $s + r$ consecutive peers. First, we define a binary-vector $(b_i, b_{i+1}, \ldots, b_{i+s+r-1})$ for each S_i, where the elements of this vector represent the state of peers of S_i: $b_j = 1$ if the peer numbered j is failed, $b_j = 0$ otherwise, $i \leq j < i+s+r$. Peer numbered $N+k$ is really the peer numbered k. Remark that the binary-vector of S_{i+1} is $(b_{i+1}, \ldots, b_{i+s+r})$.

As an example, consider a system composed of $N = 10$ peers with the values $s = 3$ and $r = 2$. The first sequence S_1 of peers is associated with the vector $(b_1, \ldots, b_5)$. If $\sum_{i=1}^{5} b_i \geq 3$, then it means that there is a data loss. Otherwise we have for example the vector $(0, 0, 1, 0, 0)$. Thus we now look at the vector $(b_2, \ldots, b_6)$ associated with the second sequence S_2 of peers. To get this new vector, we remove the first bit b_1 of the previous vector and we add the new bit b_6 at the end. We get for example $(0, 1, 0, 0, 1)$ if $b_6 = 1$. Two peer failures appear in the sequence S_2, and so we do not have a data loss. If for example $b_7 = 1$, then the vector associated with S_3 is $(1, 0, 0, 1, 1)$. In that case a data loss is found.

We now want to compute the probability to find at least one "bad" sequence S_i containing at least $r+1$ bits 1 in its vector. We use a discrete time discrete space Markov chain to represent the transitions between sequences. Indeed, the set of states V of such Markov chain is the set of all possible binary-vectors of size $s+r$ such that the sum of its

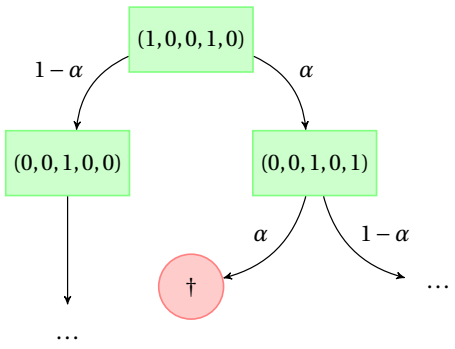


Fig. 3. Sample part of the Markov chain for $s + r = 5$ and $r + 1 = 3$.

elements is at most r, plus an absorbing state namely v_{dead} (containing all other binary-vectors of size $s + r$ in which the sum of its elements is greater than r). For a binary-vector $(b_i, b_{i+1}, \ldots, b_{i+s+r-1})$, we have two possible transitions: $(b_{i+1}, \ldots, b_{i+s+r-1}, 1)$ with probability α and $(b_{i+1}, \ldots, b_{i+s+r-1}, 0)$ with probability $1-\alpha$. One of these vectors (states) could belong to v_{dead}. Remark that we can see this Markov chain as a De Bruijn graph [5].

Consider the previous example with $s = 3$ and $r = 2$. Figure 3 describes the two possible transitions from the state $(1, 0, 0, 1, 0)$ (corresponding to the current sequence S_i): the last peer of the next sequence S_{i+1} is failed with probability α, and it is not failed with probability $1 - \alpha$. The two possible states are $(0, 0, 1, 0, 1)$ and $(0, 0, 1, 0, 0)$, respectively. Furthermore from state $(0, 0, 1, 0, 1)$, it is possible to transit to state v_{dead} because with probability α the vector of the next sequence is $(0, 1, 0, 1, 1)$.

First, we assume that the N peers are ordered in a *line* instead of a *ring*. In other words we do not take into cosideration such vectors of sequences: $(\ldots, b_N, b_1, \ldots)$. In that case we look at $N - (s + r) + 1$ sequences. We compute the distribution of probability π after N steps as follows: $\pi = v_0 M^N$ where $v_0 = (0, 0, \ldots, 0)$ is the state without peer failures and M is the transition matrix of our Markov chain. In that case $\mathbb{P}_{line}$ is $\pi(v_{dead})$.

To get the value $\mathbb{P}_{chain}$, we have to carefully take into consideration sequences containing peers on both borders of the network (becoming a ring again). The concerned sequences admit vectors $(\ldots, b_N, b_1, \ldots)$. We get $\pi = \sum_{v \in V} P(v)(v_0 M_{b_{i_1}} \ldots M_{b_{i_{s+r}}} M^{N-(s+r)}$ $M_{b_{i_1}} \ldots M_{b_{i_{s+r-1}}})$ with $P(v)$ the probability to have v as initial state, and M_k, $k \in \{0, 1\}$, the transition matrix replacing α by k.

The number of states of the previously described Markov chain is $|V| = 1 + \sum_{i=0}^{r} \binom{s+r}{i}$ states. Lemma 5 proves that we can reduce this number of states showing some properties.

Lemma 5. *There exists a Markov chain having the same $\pi(v_{dead})$ such that:*

$$|V| = 1 + \sum_{i=0}^{r} \binom{s+r}{i} - \sum_{k=1}^{r} \sum_{j=0}^{k-1} \binom{s+k-1}{j} \tag{8}$$

Proof. One of the peer failures in the chain is meaningful if and only if it can be present in some following chain containing at least $r + 1$ failures. For example, in the state $(1, 0, \ldots, 0)$, the first dead is not meaningful because, even if we have r dead peers

following, it will be too far away to make a chain with $r+1$ peer failures. In this sense, states $(0,0,\ldots,0)$ and $(1,0,\ldots,0)$ are equivalent and we can merge them.

More generally suppose we have k peer failures in the current state (current sequence of peers): we miss $r+1-k$ peer failures to make a data loss; hence, a peer failure in the current sequence will have incidence if and only if it is one of the last $s+k-1$ peers of the chain: otherwise, even if the next $r+1-k$ peers are dead, they won't fit with our k deads in a frame of size $s+r$.

Thus, among all the states with k peer failures, only those where all failures are in the tail of size $s+k-1$ are meaningful. As to the others, the first failures do not matter and we can forget them. This merging algorithm leads us to state space size (8): in a nutshell, we forget all states with k failures and less than k peer failures in the tail of size $s+k-1$.

We presented a method to compute the exact value of $\mathbb{P}_{chain}$ ($\text{MTTDL}_{chain} = 1/\mathbb{P}_{chain}$). We now propose a simple method to approximate the MTTDL using Absorbing Markov chains techniques. We first consider that the number of peers is infinite. In fact peers numbered i, $i+N$, $i+2N$, ..., $i+kN$, ... represent the same peer numbered i but at different time steps. Then the corresponding fundamental matrix gives us the average time t_{abs} to absorption, that is the average number of consecutive sequences of peers to find a data loss. Thus $MTTDL_{chain} \approx \lfloor t_{abs}/N \rfloor$. Indeed let P and Q denote the transition matrices of respectively the complete chain (described before) and the sub-chain where we removed the absorbing state and all its incident transitions. Then the fundamental matrix $R = (I-Q)^{-1}$ gives us the time to absorption t_{abs} starting from any state (see [9] for details). t_{abs} is not exactly the MTTDL since $N-(s+r)$ steps correspond to one time step (we survey the whole ring). Hence, $\lfloor t_{abs}/N \rfloor$ gives us the expected number of *time* steps before we reach the absorbing state, which is, this time, the MTTDL we are looking for.

5.2 Analytical Approximation

In the rest of this section, a *syndrome* is a sequence of $s+r$ consecutive peers containing at least $r+1$ peer failures. Under the assumption that α is "small enough" (we will see how much), we can derive an analytical expression of the MTTDL.

$$MTTDL_{chain} \approx \frac{1}{N\frac{r+1}{s+r}\binom{s+r}{r+1}\alpha^{r+1}}. \tag{9}$$

Let us begin with two lemmas.

Lemma 6. *The probability to have two distinct syndromes is negligible compared to the probability to have only one and bounded by*

$$\mathbf{P}\left[\exists \text{ two distinct syndromes } / \exists \text{ a syndrome}\right] < \frac{\alpha N(s+r)\cdot(\alpha(s+r))^{r-1}}{r!} \tag{10}$$

Proof. The probability for a syndrome to begin at a given peer (the beginning of a syndrome being considered as his first peer failure) is given by $p = \alpha\sum_{i=r}^{s+r-1}\binom{s+r-1}{i}\alpha^i(1-$

$\alpha)^{s+r-1-i}$. Meanwhile, we have

$$\mathbf{P}\left[\exists \text{ 2 distinct syndromes}\right] = \mathbf{P}\left[\cup_{|i-j|\geq s+r} \exists \text{ 2 syndromes beginning at peers } i \text{ and } j\right],$$

which is $\leq \binom{N}{2}p^2 < (pN)^2$. Normalizing by pN gives us the probability to have two syndromes knowing that there is at least one:

$$\mathbf{P}\left[\exists \text{ two distinct syndromes} \mid \exists \text{ a syndrome}\right] < pN.$$

Hence, we would like to show that pN is negligible. An upper bound on p is easy to figure out: given that $\alpha(s+r) \ll 1$, we have $p \approx \binom{s+r-1}{r}\alpha^r(1-\alpha)^{s-1} \leq (\alpha(s+r))^r/r!$, and so $pN \leq (\alpha N(s+r))(\alpha(s+r))^{r-1}/r!$. Hence, assuming $\alpha N(s+r) \ll 1$ (or otherwise $r \geq \log N$) suffices to conclude.

Lemma 7. *The probability to have more than $r+1$ dead peers in a given syndrome is negligible and bounded by*

$$\mathbf{P}\left[\exists > r+1 \textit{ dead peers} \,/\, \exists \geq r+1 \textit{ peers}\right] < \alpha(s+r) \tag{11}$$

Proof. Since we are working in a syndrome, the probability we want to bound is, in a given chain:

$$\begin{aligned}\mathbf{P}\left[\exists > \text{r+1 dead peers} \mid \exists \geq \text{r+1 dead peers}\right] &= \frac{\sum_{r+2}^{s+r}\binom{s+r}{i}\alpha^i(1-\alpha)^{s+r-i}}{\sum_{r+1}^{s+r}\binom{s+r}{i}\alpha^i(1-\alpha)^{s+r-i}} \\ &\leq \frac{\sum_{r+2}^{s+r}\binom{s+r}{i}\alpha^i(1-\alpha)^{s+r-i}}{\binom{s+r}{r+1}\alpha^{r+1}(1-\alpha)^{s-1}}\end{aligned}$$

Since the ratio between a term of the binomial series and its predecessor is $\frac{\alpha}{1-\alpha}\cdot\frac{s+r-i}{i+1}$, we can bound the tail of the binomial sum by a geometric series of common ratio $q = \frac{\alpha}{1-\alpha}\cdot\frac{s-1}{s+r} \ll 1$. Thus we have:

$$\mathbf{P}\left[\exists > \text{r+1 dead peers} \mid \exists \geq \text{r+1 dead peers}\right] < \frac{\alpha}{1-\alpha}\cdot\frac{s-1}{r+2}\cdot\frac{1}{1-q} < \alpha(s+r) \ll 1.\square$$

Therefore, if we only look for a single syndrome with exactly $r+1$ dead peers, we get a close approximation of the MTTDL.

$$\begin{aligned}\mathbb{P}_{chain} &= \mathbf{P}\left[\exists \text{ one syndrome}\right] \\ &= \mathbf{P}\left[\cup_i \exists \text{ one syndrome beginning at peer } i\right] \\ &= (N-(s+r))p\end{aligned}$$

Indeed, since there is only one syndrome, the events [syndrome begins at peer i] are exclusives. Here p is the probability for the syndrome to begin at a given peer, which we saw in proof of lemma 6. Given lemma 7, we can approximate it by $\binom{s+r-1}{r}\alpha^{r+1}(1-\alpha)^{s-1}$, which leads us too:

$$\text{MTTDL}'_{chain} \approx \frac{1}{N\binom{s+r-1}{r}\alpha^{r+1}} \tag{12}$$

One may notice that this is the same formula as (2) in the Buddy case with $c = N\frac{r+1}{s+r}$.

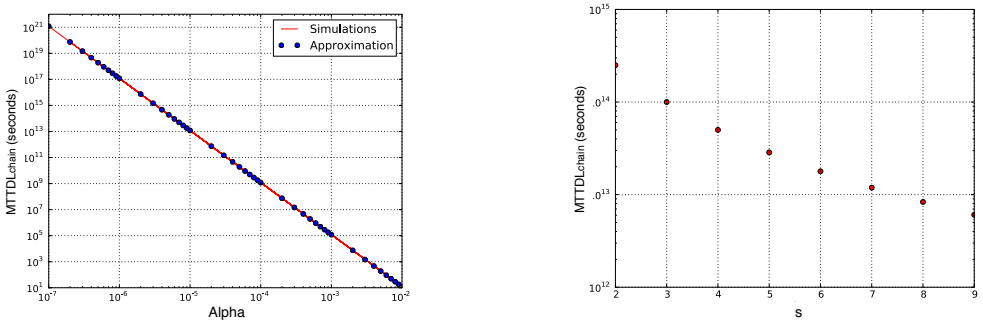


Fig. 4. Behavior of the MTTDL_{chain} when varying α (left) and s (right).

Behavior of the MTTDL Simulations led with common values of the parameters ($\alpha = 10^{-5}$, $s = 7$, $r = 3$) suggest that approximation (12) succeeds in describing the behavior of the MTTDL, as e.g. depicted by Figure 4.

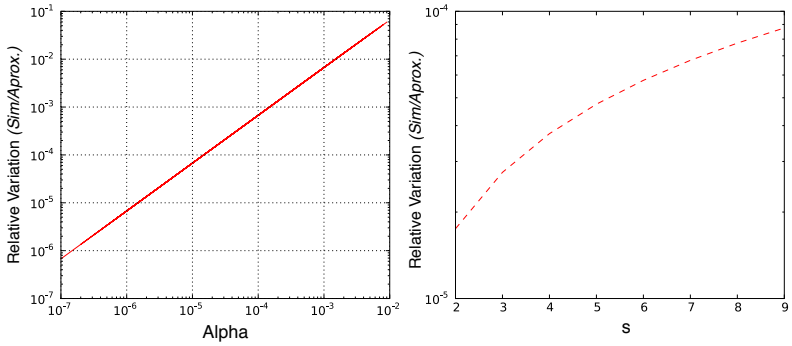


Fig. 5. Impact of α and s on the relative variation between simulation and approximation of the MTTDL_{chain}.

Validity of the approximation We have been able to compare the approximation with the exact results given by the MCM in cases where space size (8) was low enough (roughly $s < 15$ and $r < 5$), see Figure 5 for sample values. Numerical results suggested formula (12) was a good approximation for $\alpha < 10^{-3}$, s having little influence (and r almost none) on the relative variation between simulation and approximation.

6 Discussion and Conclusion

The approximations given by the Equations (2), (7), and (9) give an interesting insight on the relation between the placement policies. For instance, note that the ratio between MTTDL_{buddy} and MTTDL_{chain} does not depend of N, nor B, nor s. When $B \ll \binom{N}{r+1}$, the ratio between MTTDL_{buddy} and MTTDL_{global} depends on the number of fragments per disk $B(s+r)/N$.

$$\frac{\text{MTTDL}_{buddy}}{\text{MTTDL}_{chain}} \approx r+1, \quad \frac{\text{MTTDL}_{buddy}}{\text{MTTDL}_{global}} \approx \frac{B(s+r)}{N}, \quad \frac{\text{MTTDL}_{chain}}{\text{MTTDL}_{global}} \approx \frac{B(s+r)}{N(r+1)}.$$

We succeeded in quantifying the MTTDL of the three policies. The Buddy policy has the advantage of having a larger MTTDL than the Chain and the Global. However, when a failure occurs a large number of reconstructions start. When the bandwidth available for reconstruction is low, the reconstructions are delayed which may lead to an increased failure rate. This trade-off has still to be investigated.

References

1. R. Bhagwan, K. Tati, Y. chung Cheng, S. Savage, and G. M. Voelker. Total recall: System support for automated availability management. In *Proc. of NSDI*, pages 337–350, 2004.
2. S. Caron, F. Giroire, D. Mazauric, J. Monteiro, and S. Pérennes. P2P Storage Systems: Data Life Time for Different Placement Policies. Research Report RR-7209, INRIA, Feb 2010. http://hal.inria.fr/inria-00458190/en/.
3. B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of the NSDI'06*, pages 45–58, Berkeley, CA, USA, 2006. USENIX Association.
4. F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *Proc. of NSDI*, pages 85–98, San Francisco, USA, 2004.
5. N. De Bruijn. A combinatorial problem. *Kibern. Sb., Nov. Ser.*, 6:33–40, 1969.
6. J. R. Douceur and R. P. Wattenhofer. Large-scale simulation of replica placement algorithms for a serverless distributed file system. In *Proc. of MASCOTS*, pages 311–319, 2001.
7. S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. *19th ACM Symposium on Operating Systems Principles*, October 2003.
8. F. Giroire, J. Monteiro, and S. Pérennes. P2p storage systems: How much locality can they tolerate? In *Proc. of LCN'09*, pages 320–323, Oct 2009.
9. C. M. Grinstead and L. J. Snell. *Grinstead and Snell's Introduction to Probability*. American Mathematical Society, version dated 4 july 2006 edition, 2006.
10. S. Ktari, M. Zoubert, A. Hecker, and H. Labiod. Performance evaluation of replication strategies in dhts under churn. In *MUM '07*, pages 90–97, New York, USA, 2007. ACM.
11. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, et al. OceanStore: an architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.
12. Q. Lian, W. Chen, and Z. Zhang. On the impact of replica placement to the reliability of distributed brick storage systems. In *Proc. of ICDCS'05*, volume 0, pages 187–196, 2005.
13. A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP*, pages 188–201, 2001.
14. H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. of IPTPS*, volume 2, pages 328–338. Springer, 2002.

5.3 Configurations bien équilibrées pour le placement de données

Nous étudions les placements de données dans les systèmes avec réplication minimisant la variance du nombre de données indisponibles. Nous présentons nos résultats dans le rapport de recherche qui suit [BJMMY11].

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Well Balanced Designs for Data Placement

Jean-Claude Bermond — Alain Jean-Marie — Dorian Mazauric — Joseph Yu

N° 7725

October 2011

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7725--FR+ENG

Well Balanced Designs for Data Placement

Jean-Claude Bermond* , Alain Jean-Marie† , Dorian Mazauric† , Joseph Yu†

Thème COM — Systèmes communicants
Projets Mascotte

Rapport de recherche n° 7725 — October 2011 — 21 pages

Abstract: The problem we consider in this article is motivated by data placement in particular data replication in video on demand systems. We are given a set V of n servers and b files (data, documents). Each file is replicated on exactly k servers. A placement consists in finding a family of b subsets of V (representing the files) called blocks each of size k. Each server has some probability to fail and we want to find a placement which minimizes the variance of the number of available files. It was conjectured that there always exists an optimal placement (with variance better than that of any other placement for any value of the probability of failure). We show that the conjecture is true, if there exists a well balanced design, that is a family of blocks, such that each j-element subset of V , $1 \leq j \leq k$, belongs to the same or almost the same number of blocks (difference at most one). The existence of well balanced designs is a difficult problem as it contains as subproblem the existence of Steiner systems. We completely solve the case $k = 2$ and give bounds and constructions for $k = 3$ and some values of n and b.

Key-words: data placement, balanced designs, Steiner systems

This work was partially funded by Région PACA.

* MASCOTTE, INRIA, I3S (CNRS, Univ. Nice-Sophia Antipolis), France. firstname.lastname@inria.fr
† MAESTRO, INRIA and LIRMM, Univ. Montpellier 2, France. ajm@lirmm.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Configurations bien équilibrées pour le placement de données dans les réseaux pair-à-pair

Résumé : Nous considérons dans cet article un problème motivé par la réplication de données dans des systèmes de vidéo à la demande. Étant donnés un ensemble V de serveurs et un ensemble b de fichiers (données, documents), chaque fichier est répliqué sur exactement k serveurs. Un placement consiste à trouver une famille de b sous ensembles de V (représentant les fichiers) de taille k appelés blocs. Chaque serveur a une certaine probabilité de tomber en panne et notre but est de trouver un placement qui minimise la variance du nombre de fichiers disponibles. Il est conjecturé qu'il existe toujours un placement optimal (avec une variance meilleure que celle de n'importe quel autre placement et ce quelle que soit la probabilité de panne d'un serveur). Nous montrons que cette conjecture est vraie s'il existe une configuration équilibrée, c'est-à-dire une famille de blocs telle que tout sous ensemble à j éléments de V, $1 \leq j \leq k$, appartienne quasiment au même nombre de blocs (différence au plus 1). L'existence de configurations équilibrées est un problème difficile car un cas particulier est l'existence de systèmes de Steiner. Nous résolvons complètement le cas $k = 2$ et donnons des bornes et constructions pour $k = 3$ pour certaines valeurs de n et b.

Mots-clés : placement de données, configurations équilibrées, systèmes de Steiner

1 Introduction

The problem we consider in this article is motivated by data placement in particular data replication in video on demand systems (see [BBJMR08, BBJMR09a, BBJMR09b, JMRBB09]). We use here the terminology of design and graph theory (so the notations are somewhat different from the papers mentioned above). We are given a set V of n servers and b files (data, documents). Each file is replicated (placed) on exactly k servers. The set of servers containing file i is therefore a subset of size k, which will be called a block and denoted B_i. A placement consists of giving a family $\mathcal{F}$ of blocks $B_i, 1 \leq i \leq b$.

A server is available (on-line) with some probability δ and so unavailable (offline, failed) with the probability $1-\delta$. The file i is said to be available if one of the servers containing it is available or equivalently the file is unavailable if all the servers containing it are unavailable. In [BBJMR09a, BBJMR09b, JMRBB09] the authors study the random variable Λ, the number of available files and they proved that the mean $E(\Lambda) = b(1-(1-\delta)^k)$; so the mean is independent of the placement. However they proved that the variance of Λ depends on the placement and showed (see [JMRBB09]) that minimizing the variance corresponds to minimizing the polynomial $P(\mathcal{F}, x) = \sum_{j=0}^{k} v_j x^j$ where $x = \frac{1}{1-\delta}$ (so $x \geq 1$) and v_j denotes the number of ordered pairs of blocks intersecting in exactly j elements. So we can summarize our problem as follows:

Problem: Let n, k, b be given integers and x be a real number, $x \geq 1$; find a placement that is a family $\mathcal{F}$ of b blocks, each of size k, on a set of n elements, which minimizes the polynomial $P(\mathcal{F}, x) = \sum_{j=0}^{k} v_j x^j$, where v_j denotes the number of ordered pairs of blocks intersecting in exactly j elements. Such a placement will be called optimal for the value x.

In [JMRBB09] it is conjectured that for any n, k, b there exists a family $\mathcal{F}^*$ which is optimal for all the values of $x \geq 1$ (that is $P(\mathcal{F}^*, x) \leq P(\mathcal{F}, x)$ for any $\mathcal{F}$ and any $x \geq 1$).

Before stating our results let us give some examples. Let $n = 4, b = 4, k = 2$. We can consider different placements:

- Family $\mathcal{F}_1$: $B_1 = B_2 = B_3 = B_4 = \{1, 2\}$; then $P(\mathcal{F}_1, x) = 12x^2$
- Family $\mathcal{F}_2$: $B_1 = B_2 = \{1, 2\}, B_3 = B_4 = \{3, 4\}$; then $P(\mathcal{F}_2, x) = 4x^2 + 8$
- Family $\mathcal{F}_3$: $B_1 = \{1, 2\}, B_2 = \{1, 3\}, B_3 = \{1, 4\}, B_4 = \{2, 3\}$; then $P(\mathcal{F}_3, x) = 10x+2$
- Family $\mathcal{F}_4$: $B_1 = \{1, 2\}, B_2 = \{2, 3\}, B_3 = \{3, 4\}, B_4 = \{1, 4\}$; then $P(\mathcal{F}_4, x) = 8x+4$.

For any $x \geq 1$, $P(\mathcal{F}_4, x) \leq P(\mathcal{F}_i, x)$ and it can be proven that indeed $\mathcal{F}_4$ is an optimal family for any $x \geq 1$. Note that according to the values of x, $\mathcal{F}_2$ can be better (or worse) than $\mathcal{F}_3$. For $x \leq \frac{3}{2}$, $P(\mathcal{F}_2, x) \leq P(\mathcal{F}_3, x)$ (for example for $x = \frac{5}{4}$, $P(\mathcal{F}_2, \frac{5}{4}) = 14 + \frac{1}{4}$ and $P(\mathcal{F}_3, \frac{5}{4}) = 14 + \frac{1}{2}$). But for $x \geq \frac{3}{2}$, $P(\mathcal{F}_2, x) \geq P(\mathcal{F}_3, x)$ (for example for $x = 2$, $P(\mathcal{F}_2, 2) = 24$ and $P(\mathcal{F}_3, 2) = 22$).

Let now $n = 5, b = 3, k = 3$. We claim that the family $\mathcal{F}^*$ consisting of the three blocks $\{1,2,3\},\{1,2,4\},\{3,4,5\}$ is optimal for all $x \geq 1$. We have $P(\mathcal{F}^*, x) = 2x^2 + 4x$. Let $\mathcal{F}$ be any other family with a polynomial $P(\mathcal{F}, x) = \alpha x^3 + \beta x^2 + \gamma x + \delta$. As $n = 5$, there can never be two disjoint blocks; so $\delta = 0$. Furthermore we always have $\alpha+\beta+\gamma = b(b-1) = 6$. So $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) = (x-1)(\alpha x^2 + (\alpha+\beta-2)x)$. If $\alpha \geq 2$ (that is at least one block repeated), then $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) > 0$ for any $x > 1$. If $\alpha = 0$, among 3 blocks necessarily two of them have a pair in common and so $\beta \geq 2$ and $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) \geq 0$ for any $x > 1$.

2 Our results

For a family $\mathcal{F}$ let $\lambda^{\mathcal{F}}_{x_1,\ldots,x_j}$ (or shortly $\lambda_{x_1,\ldots,x_j}$) denote the number of blocks of the family containing the j-element subset $\{x_1,\ldots,x_j\}$. We first show that $P(\mathcal{F}, x) = \Sigma_{j=1}^{k} \Sigma_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j} (x-1)^j - bx^k + b^2$. A family $\mathcal{F}$ is j-balanced if the $\lambda_{x_1,\ldots,x_j}$ are all equal or almost equal that is if for any two j-element subsets $\{x_1,\ldots,x_j\}$ and $\{y_1,\ldots,y_j\}$, $|\lambda_{x_1,\ldots,x_j} - \lambda_{y_1,\ldots,y_j}| \leq 1$. Finally the family $\mathcal{F}$ is well balanced if it is j-balanced for $1 \leq j \leq k$. The form of the above polynomial enables us to prove that a well balanced family is also optimal and therefore the conjecture is proven for the values of b, for which there exists a well balanced family. The rest of the paper is devoted to construct well balanced families and so optimal ones. We consider first the case $k = 2$ where such families are easy to construct for any b. Then, we partly deal with the case $k = 3$ using results of design theory. Indeed the problem of constructing well balanced families contains as subproblem the existence of Steiner systems. Recall that a t-Steiner system satisfies (see [CM06]) the fact that, for any $1 \leq j \leq t$, $\lambda_{x_1,\ldots,x_j}$ is a constant. As example a $(n, 3, 1)$ Steiner triple system is defined as a family of triples (blocks of size 3) such that every pair of elements belongs to exactly one block ($\lambda_{x_1,x_2} = 1$). So it is 2-balanced; it is well known that every element belongs to exactly $\frac{n-1}{2}$ blocks and therefore it is well balanced. Such a design exists if and only if $n \equiv 1$ or $3 \pmod 6$. In that case $b = \frac{n(n-1)}{6}$. That gives some sporadic values for which there exist well balanced families. We construct many other families; as example we show that such families exist for any b for the values of $n \equiv 3 \pmod 6$ for which there exist a large number of disjoint Kirkman triple systems (see []). We deal also with other congruences of n.

3 Computation of $P(\mathcal{F}, x)$

Recall that $\lambda_{x_1,\ldots,x_j}$ denotes the number of blocks of the family containing the j-element subset $\{x_1,\ldots,x_j\}$. By convention $\lambda_\emptyset = b$.

Proposition 1. $P(\mathcal{F}, x) = \sum_{j=0}^{k} \sum_{x_1,\ldots,x_j} \lambda_{x_1,\ldots,x_j} (\lambda_{x_1,\ldots,x_j} - 1)(x-1)^j$

Proof. $P(\mathcal{F}, x) = \sum_{h=0}^{k} v_h x^h$. Let us write $P(\mathcal{F}, x) = \sum_{j=0}^{k} \mu_j (x-1)^j$. Using $x^h = (x-1+1)^h = \sum_{j=0}^{h} \binom{h}{j} (x-1)^j$, we get $\mu_j = \sum_{h=j}^{k} \binom{h}{j} v_h$.
We claim that $\mu_j = \sum_{x_1,\ldots,x_j} \lambda_{x_1,\ldots,x_j} (\lambda_{x_1,\ldots,x_j} - 1)$.

Indeed $\lambda_{x_1,\ldots,x_j}(\lambda_{x_1,\ldots,x_j}-1)$ counts the number of ordered pairs of blocks which contain $x_1,\ldots,x_j$. This number is the sum of the ordered pairs of blocks which intersect in exactly the j elements $x_1,\ldots,x_j$, plus those intersecting in exactly $j+1$ elements containing $x_1,\ldots,x_j$, plus more generally those intersecting in exactly in h elements containing $x_1,\ldots,x_j$, where, $j+2\leq h\leq k$. When we sum on all the possible j-element subsets, we therefore get

- the number of ordered pairs of blocks intersecting in exactly j elements, that is v_j
- plus the number of ordered pairs of blocks intersecting in exactly $j+1$ elements, which are counted $\binom{j+1}{j}$ times. Indeed, if the intersection of two blocks is $\{x_1,\ldots,x_{j+1}\}$ they are counted for all the j-element subsets included in $\{x_1,\ldots,x_{j+1}\}$ which are in number $\binom{j+1}{j}$. Therefore we have $\binom{j+1}{j}v_{j+1}$ such ordered pairs of blocks.
- plus for a general $h, j+2\leq h\leq k$ we count $\binom{h}{j}v_h$ ordered pairs of blocks intersecting in exactly h elements; indeed if the intersection of two blocks is $\{x_1,\ldots,x_h\}$ they are counted for all the j-element subsets included in $\{x_1,\ldots,x_h\}$ which are in number $\binom{h}{j}$.

□

We will use intensively the following equality

$$\sum_{x_1,\ldots,x_j}\lambda_{x_1,\ldots,x_j}=b\binom{k}{j} \tag{1}$$

It follows from the fact that a given block B is counted once in all the $\lambda_{x_1,\ldots,x_j}$ such that $\{x_1,\ldots,x_j\}\subset B$ and we have $\binom{k}{j}$ such j-element subsets.

Theorem 1. $P(\mathcal{F},x)=\sum_{j=1}^{k}\sum_{x_1,\ldots,x_j}\lambda^2_{x_1,\ldots,x_j}(x-1)^j-bx^k+b^2$

Proof. Using equation 1, we get $\sum_{j=0}^{k}\sum_{x_1,\ldots,x_j}\lambda_{x_1,\ldots,x_j}(x-1)^j=\sum_{j=0}^{k}b\binom{k}{j}(x-1)^j=bx^k$. Replacing in the expression of $P(\mathcal{F},x)$ given in Proposition 1 and using the fact that $\lambda^2_\emptyset=b^2$ we obtain the theorem. □

4 Well balanced families

A family $\mathcal{F}$ is j-balanced if all the $\lambda_{x_1,\ldots,x_j}$ are equal or almost equal that is if for any two j-element subsets $\{x_1,\ldots,x_j\}$ and $\{y_1,\ldots,y_j\}$, $|\lambda_{x_1,\ldots,x_j}-\lambda_{y_1,\ldots,y_j}|\leq 1$. Finally the family $\mathcal{F}$ is well balanced if it is j-balanced for $1\leq j\leq k$.

Proposition 2. $\sum_{x_1,\ldots,x_j}\lambda^2_{x_1,\ldots,x_j}$ *is minimized when* $\mathcal{F}$ *is j-balanced.*

Proof. As by equation 1, $\sum_{x_1,\ldots,x_j} \lambda_{x_1,\ldots,x_j}$ is a constant, $\sum_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j}$ is minimized when all the $\lambda_{x_1,\ldots,x_j}$ are equal to $b\frac{\binom{k}{j}}{\binom{n}{j}}$ if this value is an integer or equal either to $\lfloor b\frac{\binom{k}{j}}{\binom{n}{j}} \rfloor$ or $\lceil b\frac{\binom{k}{j}}{\binom{n}{j}} \rceil$. That is equivalent to say that $\mathcal{F}$ is j-balanced.

□

So, we can state our main theorem

Theorem 2. *If $\mathcal{F}^*$ is well balanced, then $\mathcal{F}^*$ is optimal that is $P(\mathcal{F}^*, x) \leq P(\mathcal{F}, x)$ for any $\mathcal{F}$ and any $x \geq 1$.*

Proof. If $\mathcal{F}^*$ is well balanced, then all the coefficients of the polynomial as expressed in the Theorem 1 are minimized and so $\mathcal{F}^*$ is optimal. □

Note that for a j-balanced family, the coefficient of $(x-1)^j$ in the polynomial $P(\mathcal{F}, x)$ is easy to compute. Let $b\binom{k}{j} = q\binom{n}{j} + r$, then we have r values of the $\lambda_{x_1,\ldots,x_j}$ equal to $q+1$ and $\binom{n}{j} - r$ equal to q. So, $\sum_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j} = \binom{n}{j}q^2 + 2qr + r$.

Complete well balanced family:

If $b = \binom{n}{k}$ the complete family consisting of all the possible k-subsets is well balanced, with the values of the $\lambda_{x_1,\ldots,x_j}$ being all equal to $\lambda_j = \binom{n-j}{k-j}$. By taking h copies we get also a well balanced family for $b = h\binom{n}{k}$.

Proposition 3. *Let n and k be given and let $b' = h\binom{n}{k} + b$ with $b \leq \binom{n}{k}$. Then, there exists a well balanced family $\mathcal{F}'$ for b' if and only if there exists a well balanced family $\mathcal{F}$ for b*

Proof. If we have a well balanced family $\mathcal{F}$ for some $b \leq \binom{n}{k}$ we can construct a well balanced family $\mathcal{F}'$ for $b' = h\binom{n}{k} + b$ by adding h complete families to $\mathcal{F}$. Conversely if we have a well balanced family $\mathcal{F}'$ for $b' = h\binom{n}{k} + b$, each k-element subset is repeated h or $h+1$ times and so by deleting h copies of each block, we can deduce a well balanced family for b. □

The next proposition generalizes this idea to optimal families.

Proposition 4. *Let n and k be given and let $b' = h\binom{n}{k} + b$ with $b \leq \binom{n}{k}$. If there exists an optimal family for b', then there exists an optimal family for b and furthermore the optimal family for b' consists of the optimal family for b plus h complete families.*

Proof. Suppose there exists an optimal family $\mathcal{F}'$ for b'. This family is necessarily k-balanced. Indeed suppose it is not the case and let $\mathcal{G}'$ be a k-balanced family (such a family can be easily constructed by taking among the $\binom{n}{k}$ subsets of size k, b of them repeated $h+1$ times and the other $\binom{n}{k} - b$ repeated h times). But, the coefficient of x^k in $P(\mathcal{G}', x)$ will be strictly less than that of $P(\mathcal{F}', x)$ and so for x large enough $P(\mathcal{G}', x) < P(\mathcal{F}', x)$ contradicting the optimality of $\mathcal{F}'$. So each k-element subset appears exactly h or $h+1$ times. Deleting h

copies of each block we get a family $\mathcal{F}$ with $b = b' - \binom{n}{k}$ blocks (none of them being repeated). Note that if $\lambda_{x_1,\ldots,x_j}$ (resp $\lambda'_{x_1,\ldots,x_j}$) denotes the number of blocks of the family $\mathcal{F}$ (resp $\mathcal{F}'$) containing $\{x_1,\ldots,x_j\}$ we have : $\lambda'_{x_1,\ldots,x_j} = \lambda_{x_1,\ldots,x_j} + h\binom{n-j}{k-j}$. Consider another family $\mathcal{G}$ on b blocks and let $\mathcal{G}'$ be the family on b' blocks obtained by adding h complete families to $\mathcal{G}$. Let $\mu_{x_1,\ldots,x_j}$ (resp $\mu'_{x_1,\ldots,x_j}$) denote the number of blocks of the family $\mathcal{G}$ (resp $\mathcal{G}'$) containing $\{x_1,\ldots,x_j\}$. Then we have : $\mu'_{x_1,\ldots,x_j} = \mu_{x_1,\ldots,x_j} + h\binom{n-j}{k-j}$. So, as by equation 1, $\sum_{x_1,\ldots,x_j} \lambda_{x_1,\ldots,x_j} = \sum_{x_1,\ldots,x_j} \mu_{x_1,\ldots,x_j}$ and $\sum_{x_1,\ldots,x_j} \lambda'_{x_1,\ldots,x_j} = \sum_{x_1,\ldots,x_j} \mu'_{x_1,\ldots,x_j}$, then $\sum_{x_1,\ldots,x_j} \lambda^2_{x_1,\ldots,x_j} - \sum_{x_1,\ldots,x_j} \mu^2_{x_1,\ldots,x_j} = \sum_{x_1,\ldots,x_j} \lambda'^2_{x_1,\ldots,x_j} - \sum_{x_1,\ldots,x_j} \mu'^2_{x_1,\ldots,x_j}$ and thus $P(\mathcal{G}',x) - P(\mathcal{F}',x) = P(\mathcal{G},x) - P(\mathcal{F},x)$. Therefore if $\mathcal{F}$ is not optimal there exists a family $\mathcal{G}$ and a value x for which $P(\mathcal{G},x) < P(\mathcal{F},x)$ and for this value of x we have $P(\mathcal{G}',x) < P(\mathcal{F}',x)$ and $\mathcal{F}'$ will not have been optimal, a contradiction. □

Note that we conjecture that the converse is true: that is starting from an optimal family $\mathcal{F}$ for some $b \leq \binom{n}{k}$, the family $\mathcal{F}'$ obtained by adding h complete families is also optimal. That is verified, if the conjecture of [JMRBB09] on the existence of an optimal family for any n, b, k is true as in that case any optimal family is k-balanced.

In what follows we will restrict ourselves to the case $b \leq \binom{n}{k}$. In fact the following proposition shows that we can consider only the values of $b \leq \frac{1}{2}\binom{n}{k}$.

Proposition 5. *Let n and k be given, an optimal family $\bar{\mathcal{F}}$ for $\bar{b} = \binom{n}{k} - b$ is obtained from an optimal family $\mathcal{F}$ for $b \leq \binom{n}{k}$ by taking as blocks the k-subsets which are not blocks of $\mathcal{F}$.*

Proof. Let $\mathcal{F}$ be an optimal family with b blocks and let $\bar{\mathcal{F}}$ be the family obtained from $\mathcal{F}$ by taking as blocks the k-subsets which are not blocks of $\mathcal{F}$. $\bar{\mathcal{F}}$ has $\bar{b} = \binom{n}{k} - b$ blocks. Furthermore, if $\bar{\lambda}_{x_1,\ldots,x_j}$ denotes the number of blocks of the family $\bar{\mathcal{F}}$ containing $\{x_1,\ldots,x_j\}$, we have $\bar{\lambda}_{x_1,\ldots,x_j} = \binom{n-j}{k-j} - \lambda_{x_1,\ldots,x_j}$. Consider another family $\bar{\mathcal{G}}$ with $\bar{b}$ blocks and let $\mathcal{G}$ be the complementary family obtained from $\bar{\mathcal{G}}$ by taking as blocks the k-subsets which are not blocks of $\bar{\mathcal{G}}$; $\mathcal{G}$ has b blocks. We also have: $\bar{\mu}_{x_1,\ldots,x_j} = \binom{n-j}{k-j} - \mu_{x_1,\ldots,x_j}$ and so we get $P(\bar{\mathcal{G}},x) - P(\bar{\mathcal{F}},x) = P(\mathcal{G},x) - P(\mathcal{F},x)$. Therefore if $\mathcal{F}$ is an optimal family, then $\bar{\mathcal{F}}$ is also an optimal family.

□

Well balanced families and Steiner systems:

Recall that a t-Steiner system (or (n,k,λ) t-design) is a family of blocks such that each t-element subset appears in exactly λ blocks (see [CM06, CJ06]). In that case it is well known that also, for $1 \leq j \leq t$ each j-element subset appears in exactly λ_j blocks, where $\lambda_j = \frac{\lambda}{t+1-j}\binom{n-j}{t-j}$. So a t-design is j-balanced for all j, $1 \leq j \leq t$. In particular, if $t = k-1$ and the blocks are repeated the same or almost the same number of times, then a k-Steiner family is also well balanced. As an example, a Steiner Triple System (STS) consists of a

family of triples, such that each pair of elements appears in exactly one triple. In that case each element appears in $\frac{n-1}{2}$ triples and no triple is repeated. Therefore, a STS is a well balanced family. It is well known that a STS exists if and only if $n \equiv 1$ or 3 mod (6) and then $b = \frac{n(n-1)}{6}$.

5 Case k = 2

Theorem 3. *Let $k = 2$. Then for any n and b there exists a well balanced family.*

Proof. We have only to consider the case $b \leq \binom{n}{2}$. In the case $k = 2$ the blocks are pairs of elements and so the problem consists of designing a simple graph with n vertices and b edges and almost regular (the degree of a vertex x being $d(x) = \lfloor \frac{2b}{n} \rfloor$ or $\lceil \frac{2b}{n} \rceil$). We distinguish two cases:

- Case n even: let $b = q\frac{n}{2} + r$ for $0 \leq r < \frac{n}{2}$. It is well known that, for n even, the edges of the complete graph K_n can be decomposed into $n-1$ perfect matchings (set of $\frac{n}{2}$ disjoint edges covering the vertices). In that case the family consisting of q perfect matchings plus r edges of the $(q+1)$th perfect matching forms the required family with $b = q\frac{n}{2} + r$ edges, none of them repeated and with the degree of a vertex equal to q or $q+1$.
- Case n odd: let $b = qn + r$ for $0 \leq r < n$. It is also well known that for n odd, the edges of complete graph K_n can be decomposed into $\frac{n-1}{2}$ hamiltonian cycles (cycles containing each vertex exactly once). In that case consider the family consisting of q hamiltonian cycles plus the following r edges of the $(q+1)$th hamiltonian cycle: if the cycle is $x_0, x_1, \ldots, x_i, \ldots, x_{n-1}$ we take the r edges $\{x_{2j}, x_{2j+1}\}$ for $0 \leq j \leq r-1$ (indices being taken modulo n). Then it consists of $b = qn + r$ edges none of them being repeated; furthermore the degree of a vertex is $2q$ or $2q+1$ if $r \leq \frac{n-1}{2}$ and $2q+1$ or $2q+2$ otherwise and so in both cases $d(x) = \lfloor \frac{2b}{n} \rfloor$ or $\lceil \frac{2b}{n} \rceil$.

□

Algorithm to construct a well balanced family starting from any family:
In some cases (files or servers appearing or disappearing), it might be helpful to design an algorithm, which starting from some family constructs an optimal well balanced family. That is in general a difficult problem; but for $k = 2$, we can easily design such a procedure. Let n and b be given and $k = 2$ and consider any family $\mathcal{F}$; we will transform it into a well balanced family with the same parameters. First let us construct a 2-balanced family. Suppose, $\mathcal{F}$ is not 2 balanced; so there exist two edges (blocks) $\{x, y\}$ and $\{z, t\}$ with $\lambda_{xy} \geq \lambda_{zt} + 2$. Then, delete from $\mathcal{F}$ one edge $\{x, y\}$ and add one edge $\{z, t\}$. Repeating this procedure we end after a finite number of steps with a family such that for any pair of edges $\{x, y\}$ and $\{z, t\}$ $\lambda_{xy} \leq \lambda_{zt} + 1$, that is a 2-balanced family.
Now let us show how to construct a well balanced family from a 2-balanced one. Let $\mathcal{F}$ be a

2-balanced family with $\lambda_{xy} = \lambda$ or $\lambda - 1$; suppose it is not 1-balanced; then there exist two vertices x and z with $d(x) \geq d(z) + 2$. So there exists a vertex $y \neq x, z$ with $\lambda_{xy} \geq \lambda_{zy} + 1$; otherwise $d(x) = \sum_{y \neq x,z} \lambda_{xy} + \lambda_{xz} \leq \sum_{y \neq x,z} \lambda_{zy} + \lambda_{xz} = d(z)$ a contradiction. So, $\lambda_{xy} = \lambda$ and $\lambda_{zy} = \lambda - 1$. Deleting from $\mathcal{F}$ one edge $\{x, y\}$ and adding one edge $\{z, y\}$, we still get a 2-balanced family $\mathcal{F}'$ ($\lambda'_{xy} = \lambda - 1$ and $\lambda'_{zy} = \lambda$); but we have reduced the gap between the degrees of x and z (as $d'(x) = d(x) - 1$ and $d'(z) = d(z) + 1$ the other degrees being unchanged). Repeating this procedure we end after a finite number of steps with a 1-balanced and 2-balanced, so well balanced family.

6 Case k = 3

The cases $k > 2$, are much more complicated. Already for $k = 3$, there are values of n and b for which there do not exist well balanced families. For $n = 4$ and $b = 2$, if there exists a 2-balanced family, then $\lambda_{xy} = 1$, but that is impossible as $n - 1 = 3$ and there cannot exist a partition of the edges of K_4 into triples (non existence of $(4, 3, 1)$-design). The argument is generalized in the following proposition :

Proposition 6. *Let $k = 3$, n be even and λ be odd. If $\lambda\frac{n(n-1)}{2} - \frac{n}{2} < 3b < \lambda\frac{n(n-1)}{2} + \frac{n}{2}$, then there does not exist a 2-balanced family.*

Proof. Note that the number of possible pairs is $\frac{n(n-1)}{2}$. We distinguish 3 cases.

- $\lambda\frac{n(n-1)}{2} = 3b$. In that case a 2-balanced family will verify $\lambda_{xy} = \lambda$ for all pairs $\{x, y\}$ and then we should have $\lambda_x = \lambda\frac{n-1}{2}$ impossible as λ is odd and n is even (non existence of an $(n, 3, \lambda$-design).
- $\lambda\frac{n(n-1)}{2} - \frac{n}{2} < 3b$. In that case we cannot have all the $\lambda_{xy} \geq \lambda$. So we have one of the $\lambda_{xy} \leq \lambda - 1$ and if the family is 2-balanced all the $\lambda_{xy} \leq \lambda$. But, then $\lambda_x \leq \lambda\frac{n-1}{2}$ and as $\lambda\frac{n-1}{2}$ is odd, $\lambda_x \leq \lambda\frac{n-1}{2} - \frac{1}{2}$. Therefore, using Equation 1, $3b = \sum_x \lambda_x \leq \lambda\frac{n(n-1)}{2} - \frac{n}{2}$ a contradiction.
- $3b < \lambda\frac{n(n-1)}{2} + \frac{n}{2}$. In that case we cannot have all the $\lambda_{xy} \leq \lambda$. So we have one of the $\lambda_{xy} \geq \lambda + 1$ and if the family is 2-balanced all the $\lambda_{xy} \geq \lambda$. But, then $\lambda_x \geq \lambda\frac{n-1}{2}$ and as $\lambda\frac{n-1}{2}$ is odd, $\lambda_x \geq \lambda\frac{n-1}{2} + \frac{1}{2}$. Therefore, using Equation 1, $3b = \sum_x \lambda_x \geq \lambda\frac{n(n-1)}{2} + \frac{n}{2}$ a contradiction.

□

Examples of application : By Proposition 6, there do not exist well balanced families for $k = 3$ and $\{n = 6; b = 5\}$; $\{n = 8; b = 9, 10, 27, 28, 29\}$; $\{n = 10; b \equiv 14, 15, 16 \pmod{30}\}$; $\{n = 16; 38, 39, 40, 41, 4242 \pmod{80}\}$.

Proposition 7. *Let $k = 3$, if $\lambda\frac{n(n-1)}{6}$ is not an integer, then there does not exist a well balanced family for $b = \lfloor\lambda\frac{n(n-1)}{6}\rfloor$ or $b' = \lceil\lambda\frac{n(n-1)}{6}\rceil$*

Proof. If $\lambda\frac{n(n-1)}{6}$ is not an integer, let $b = \lfloor \lambda\frac{n(n-1)}{6} \rfloor$; then $3b = \lambda\frac{n(n-1)}{2} - \epsilon$ where $\epsilon = 1$ or 2. By Equation 1, $3b = \sum_x \lambda_x$ and so if $\mathcal{F}$ is 1-balanced $\lambda_x = \frac{\lambda(n-1)}{2}$ except for ϵ vertices for which the value is one less. Similarly by Equation 1, $3b = \sum_x \lambda_{x,y}$ and so if $\mathcal{F}$ is 2-balanced $\lambda_{x,y} = \lambda$ except for ϵ pairs appearing $\lambda - 1$ times. But for an x_0 with $\lambda_{x_0} = \frac{\lambda(n-1)}{2} - 1$, we have $\lambda(n-1) - 2$ pairs containing it and so two pairs appear $\lambda - 1$ times. If $\epsilon = 2$ we have another vertex x_0' with $\lambda_{x_0'} = \frac{\lambda(n-1)}{2} - 1$ and altogether at least 3 pairs appear $\lambda - 1$ times (only the pair $\{x_0, x_0'\}$ can be counted twice). So we have at least $\epsilon + 1$ pairs appearing $\lambda - 1$ times, contradicting the fact that if if $\mathcal{F}$ is 2-balanced only ϵ pairs appear $\lambda - 1$ times.

The proof for $b' = \lceil \lambda\frac{n(n-1)}{6} \rceil$ is similar. $3b' = \lambda\frac{n(n-1)}{2} + \epsilon$ where $\epsilon = 1$ or 2. If $\mathcal{F}$ is 1-balanced $\lambda_x = \frac{\lambda(n-1)}{2}$ except for ϵ vertices for which the value is one more. If $\mathcal{F}$ is 2-balanced $\lambda_{x,y} = \lambda$ except for ϵ pairs appearing $\lambda + 1$ times. The argument applied for the x_0 with $\lambda_{x_0} = \frac{\lambda(n-1)}{2} + 1$ gives that at least $\epsilon + 1$ pairs appear $\lambda + 1$ times, a contradiction. □

Examples of applications : The Proposition 7 applies when $n \equiv 5 \pmod 6$ and $\lambda \not\equiv 0 \pmod 3$; for example for $\{n = 5; b = 3, 4\}$ or $\{n = 11; b = 18, 19, 36, 37\}$. It applies also for $n \equiv 2 \pmod 6$ and $\lambda \not\equiv 0 \pmod 3$; for λ odd it is included in Proposition 6, but for λ even we get new values such as $\{n = 8; b = 18, 19\}$ $\{n = 14; b = 60, 61, 121, 122\}$.

We conjecture that the values excluded by the two Propositions 6 and 7 are the only ones for which there do not exist well balanced families.

Conjecture 1. *Let $k = 3$, there exists a well balanced family for the values of n and b different from that of Propositions 6 and 7. In particular we conjecture that, if $n \equiv 1$ or 3 (mod 6), then there exists a well balanced family for any b.*

On a positive side and to try to prove the conjecture 1, we can use all the results obtained in design theory in particular on Steiner Triple Systems (see the handbook [CJ06] for details) to construct some well balanced families. Recall that a $(n, 3, 1)$ Steiner Triple System (STS(n) shortly) is defined as a family of triples (blocks of size 3), such that every pair of elements belongs to exactly one block ($\lambda_{x,y} = 1$). So it is 2 balanced; it is well known that every vertex belongs to exactly $\frac{n-1}{2}$ blocks and therefore it is well balanced. Such a design exists if and only if $n \equiv 1$ or 3 (mod 6). In that case $b = \frac{n(n-1)}{6}$.

Example 1: For $n = 7$, the blocks of $(7, 3, 1)$-design are $B_i = \{i, i+1, i+3\}$, $0 \le i \le 6$, the numbers being taken modulo 7. For $n = 9$, the blocks of an STS(9) are given in the following array.

Example 1. *Two disjoint Kirkman triple Systems for $n = 9$*

$\{0, \infty, \infty'\}$	$\{0, 2, 5\}$	$\{0, 3, 4\}$	$\{0, 1, 6\}$
$\{1, 2, 4\}$	$\{1, 3, \infty'\}$	$\{1, 5, \infty\}$	$\{2, 3, \infty\}$
$\{3, 5, 6\}$	$\{4, 6, \infty\}$	$\{2, 6, \infty'\}$	$\{4, 5, \infty'\}$

Example 1(a): *a Kirkman triple System for $n = 9$*

$\{1,\infty,\infty'\}$	$\{1,3,6\}$	$\{1,4,5\}$	$\{1,2,0\}$
$\{2,3,5\}$	$\{2,4,\infty'\}$	$\{2,6,\infty\}$	$\{3,4,\infty\}$
$\{4,6,0\}$	$\{5,0,\infty\}$	$\{3,0,\infty'\}$	$\{5,6,\infty'\}$

Example 1(b): *another Kirkman Triple System for* $n = 9$

That gives some sporadic values for which there exist well balanced families. We can get more values of b by considering more than one STS(n); but we have to insure that the family is 3-balanced (that is no block is repeated). Fortunately the answer is obtained thanks to Theorem 4. Two STS(n) are said to be disjoint if they have no triple in common. A set of $n-2$ disjoint STS(n) is called a *large set of disjoint STS(n)* and briefly denoted by LSTS(n). An LSTS(n) can be viewed as a partition of the complete family of $\binom{n}{3}$ triples into STS(n). In 1850, Cayley showed that there are only two disjoint STS(7) and so there is no LSTS(7). The same year Kirkman showed that there exists an LSTS(9). Such an LSTS(9) is given by taking as first STS(9) the STS of Example 1; the 6 other STS(9) are obtained from the first one by developing modulo 7 (that is applying the automorphism fixing ∞ and ∞' and mapping i to $i+1$). For example, the second STS(9) is obtained by adding 1 to each number (∞ and ∞' are invariant and $6+1=0 \pmod 7$) and given as Example 2.

Due to the efforts of many authors the following theorem completely settles the existence of LSTS(n).

Theorem 4. *([Lu83, Lu84, Tei91] (see [Ji05] for a simple proof) For* $n \equiv 1$ *or* 3 *(mod 6),* $n > 7$*, there exists an LSTS(n).*

Proposition 8. *Let* $k = 3$*, and* $n \equiv 1$ *or* 3 *(mod 6),* $n > 7$*, then there exists a well balanced family for any* b *multiple of* $\frac{n(n-1)}{6}$*.*

Proof. Let $b = h\frac{n(n-1)}{6}$; $b \leq \binom{n}{3}$. Then, the family consisting of h disjoint STS(n) is well balanced (with $\lambda_{xy} = h$ and $\lambda_x = h\frac{n-1}{2}$). For $b \geq \binom{n}{3}$ the result follows by using Proposition 3 □

We will see after that the existence of two disjoint STS(7) suffices to construct a well balanced family for $n = 7$ and any b.

When $n = 6t + 3$, there exist STS(n) which have a stronger property. The triples of the STS can themselves be partitionned into $3t + 1$ classes, called *parallel classes*, where a parallel class consists of $2t+1$ blocks forming a partition of the n elements. Such an STS(n) is called resolvable or a Kirkman Triple System (briefly KTS(n)). Example 1 is a KTS(9), the 4 parallel classes correspond to the 4 columns. It is well known that a KTS(n) exists for any $n \equiv 3 \pmod 6$ ([RCW71]). Two KTS(n) are said to be disjoint if they have no triple in common. A set of $n-2$ disjoint KTS(n) is called a *large set of disjoint KTS(n)* and briefly denoted by LKTS(n). In 1850, Kirkman showed that an LKTS(9) exists and in 1974, Denniston found an LKTS(15). For $n = 9$, the LSTS(9) given before, is in fact an LKTS(9) as the resolvabilty is kept by the automorphism. An example of a KTS(15)denoted K_A is

given below. Developing modulo 13, that is applying the automorphism fixing ∞ and ∞' and mapping i to $i+1$, we get 13 disjoint KTS(15) and so a LKTS(15). Example 4 shows $K_B = K_A + 1$

Example 2. *Two disjoint Kirkman triple Systems for* $n = 15$

$\{0,1,9\}$	$\{0,2,7\}$	$\{0,3,11\}$	$\{0,4,6\}$	$\{0,5,8\}$	$\{0,10,12\}$	$\{1,4,5\}$
$\{2,4,12\}$	$\{3,4,8\}$	$\{1,7,12\}$	$\{1,8,11\}$	$\{1,2,3\}$	$\{3,5,9\}$	$\{2,6,11\}$
$\{5,10,11\}$	$\{5,6,12\}$	$\{6,8,10\}$	$\{2,9,10\}$	$\{6,7,9\}$	$\{4,7,11\}$	$\{3,7,10\}$
$\{7,8,\infty\}$	$\{9,11,\infty\}$	$\{2,5,\infty\}$	$\{3,12,\infty\}$	$\{4,10,\infty\}$	$\{1,6,\infty\}$	$\{8,9,12\}$
$\{3,6,\infty'\}$	$\{1,10,\infty'\}$	$\{4,9,\infty'\}$	$\{5,7,\infty'\}$	$\{11,12,\infty'\}$	$\{2,8,\infty'\}$	$\{0,\infty,\infty'\}$

Example 2(a): *a Kirkman Triple System* K_A *for* $n = 15$

$\{1,2,10\}$	$\{1,3,8\}$	$\{1,4,12\}$	$\{1,5,7\}$	$\{1,6,9\}$	$\{1,11,0\}$	$\{2,5,6\}$
$\{3,5,10\}$	$\{4,5,9\}$	$\{2,8,0\}$	$\{2,9,12\}$	$\{2,3,4\}$	$\{4,6,10\}$	$\{3,7,12\}$
$\{6,11,12\}$	$\{6,7,0\}$	$\{7,9,11\}$	$\{3,10,11\}$	$\{7,8,10\}$	$\{5,8,12\}$	$\{4,8,11\}$
$\{8,9,\infty\}$	$\{10,12,\infty\}$	$\{3,6,\infty\}$	$\{4,0,\infty\}$	$\{5,11,\infty\}$	$\{2,7,\infty\}$	$\{9,10,0\}$
$\{4,7,\infty'\}$	$\{2,11,\infty'\}$	$\{5,10,\infty'\}$	$\{6,8,\infty'\}$	$\{12,0,\infty'\}$	$\{3,9,\infty'\}$	$\{1,\infty,\infty'\}$

Example 2(b): *another Kirkman Triple System* K_B *for* $n = 15$

Since that, many people have done some research on their existence. The more recent paper is [ZC10] where the reader can find other references. The results are summarized in the following theorem:

Theorem 5. *[ZC10] There exists an LKTS*$(3^a 5^b r \prod_{i=1}^{s}(2.13^{n_i}+1 \prod_{j=1}^{t}(2.7^{m_j}+1))$ *for any integer* $r \in \{7,13\}, n_i, m_j \geq 1 (1 \leq i \leq s, 1 \leq j \leq t), a \geq 1, b, s, t \geq 0$ *and further* $a+s+t \geq 2$ *if* $b \geq 1$.

Proposition 9. *Let* $k = 3$ *and* $n = 6t+3$. *If there exists an LKTS(n) then there exists a well balanced family for any b.*

Proof. By Proposition 3, we can suppose $b \leq \binom{n}{3}$. Let $b = q(2t+1)(3t+1) + r(2t+1) + s$ with $0 \leq q < 6t+1; 0 \leq r < 3t+1; 0 \leq s < 2t+1$. Then a well balanced family consists of q disjoint KTS(n) of an LKTS(n), plus r parallel classes of the $(q+1)$th KTS(n) and s triples of the $(r+1)$th parallel class of this KTS(n). Indeed by definition of an LKTS, all the triples are disjoint and so $\lambda_{x,y,z} = 0$ or 1. In each KTS a pair of elements appears exactly once; so $\lambda_{x,y} = q$ or $q+1$ (exactly q if $r = 0, s = 0$). In each parallel class, each vertex appears exactly once ; so $\lambda_x = (3t+1)q + r$ or $(3t+1)q + r + 1$ (exactly $(3t+1)q + r$ if $s = 0$). □

Note that, according to the proof above and the fact that by Proposition 5 we can suppose $b \leq \frac{1}{2}\binom{n}{3}$, we do not need to have a structure as strong as an LKTS, but only $3t+1$ disjoint STS, with one of them being a KTS (the one used as the $(q+1)$th KTS). We conjecture such structure always exists for $n = 6t+3$.

Conjecture 2. *For $n = 6t + 3$, there exist $3t + 1$ disjoint STS(n) one of them being a KTS(n).*

Conjecture 1 will follow from Conjecture 2 for $n \equiv 3 \pmod 6$. For $n \equiv 1 \pmod 6$ we need to have some extra property.

From Proposition 8 and Proposition 9, we can also construct well balanced families for $n-1 = 6t$ or $6t+2$ (resp. $n+1 = 6t+2$ or $6t+4$) by deleting (resp. adding) one from (to) a well balanced family for $n = 6t + 1$ or $6t + 3$; but we have to do it carefully.

Proposition 10. *Let $k = 3$ and $n = 6t$ (resp $n = 6t + 2$). There exists a well balanced family for $b = ht(6t - 2)$ (resp $b = ht(6t + 2)$).*

Proof. Take, as $n + 1 \equiv 1$ or 3 (mod 6), a set of h disjoint STS(n+1) and delete all the $h\frac{n}{2}$ blocks containing the element $n + 1$. □

We can extend this construction to other values. As example, consider $n = 8, b = 12$. We can start from the 8 blocks obtained in Proposition 10 by deleting ∞' in the Example 1 of a KTS(9). Note that $\lambda_{xy} = 1$ except for the 4 pairs $\{0, \infty\}, \{1, 3\}, \{2, 6\}, \{4, 5\}$. We can add now 4 blocks taken from another KTS(9), for example that of Example 2, containing these pairs; namely the blocks $\{5, 0, \infty\}, \{1, 3, 6\}, \{2, 6, \infty\}, \{1, 4, 5\}$.

In fact it appears more efficient to get many well balanced families for $n = 6t + 4$ (and also $n = 6t + 2$) by starting from an LKTS(6t+3) (or an LSTS(6t+1)) and adding a new element $\alpha = 6t + 4$ (or $6t + 2$).

The following construction (called Construction A) will be useful.

Construction A: Consider a parallel class of a KTS($6t+3$) and a new element α ($= 6t+4$) and replace each of the $2t+1$ triples $\{x_j, y_j, z_j\}$ of this class ($1 \le j \le 2t+1$) by the 3 triples $\{x_j, y_j, \alpha\}$, $\{x_j, z_j, \alpha\}$, and $\{y_j, z_j, \alpha\}$. As example take the KTS(9) of Example 1. We will replace the first class consisting of the 3 blocks $\{0, \infty, \infty'\}, \{1, 2, 4\}, \{3, 5, 6\}$ by the 9 blocks $\{0, \infty, \alpha\}, \{0, \alpha, \infty'\}, \{\alpha, \infty, \infty'\}$ $\{1, 2, \alpha\}, \{1, \alpha, 4\}, \{\alpha, 2, 4\}, \{3, 5, \alpha\}, \{3, \alpha, 6\}, \{\alpha, 5, 6\}$.

Proposition 11. *Let $k = 3$ and $n = 6t + 4$. If there exist $2p$ ($2p \le 6t + 1$) disjoint STS($6t + 3$) one of them being a KTS($6t + 3$), then there exists a well balanced family for $b_{2p} = 2p(3t + 2)(2t + 1)$.*

Proof. We will use construction A. Start with $q = 2p$ disjoint STS($6t + 3$) and do the construction A for p classes of the KTS denoted K_A. As the classes are taken in the same KTS, α appears in $3p(2t + 1)$ disjoint triples, where each vertex x appears exactly $2p$ times; so $\lambda_{\alpha x} = 2p$ and $\lambda_\alpha = 3p(2t+1)$. Doing so we have not changed the values of $\lambda_{xy} = 2p$, as we took $2p$ disjoint STS; but λ_x has increased by p and its value is now $2p(3t+1)+p = 3p(2t+1)$. Therefore the family constructed is well balanced. □

We can extend the theorem to get well balanced families for more values of b either by deleting or by adding blocks.

Deletion process We start from the well balanced family obtained for $b = b_{2p}$ in Proposition 11 Construction A being done in K_A for the value p. We suppose furthermore in what follows that there exists among the $2p$ STS a second disjoint KTS($6t+3$) denoted K_B. We can now delete the blocks appearing in another non modified class of K_A, but it appears better to delete the blocks appearing in a class of the KTS K_B. Doing so, we get a well balanced family for $b_{2p} - (2t+1) \leq b \leq b_{2p}$. Now we have to delete some blocks containing α. We will use the following deletion process.

Deletion A-B: This construction consists of deleting a block $\{x, y, \alpha\}$ appearing in a modified class of K_A and the blocks of a class of K_B containing $\{x, y\}$, except the block $\{x, y, z\}$. Doing so some pairs appear one less and if we do it for all the blocks of the class of K_B all elements appear one less except z.

We can do deletion A-B h times, $h \leq 3t+1$ = number of classes of K_B, if the two following conditions hold

- all the pairs $\{x, y\}$ are disjoint (in order to avoid to delete twice a pair $\{x, \alpha\}$).
- any two blocks of K_B that we keep $\{x, y, z\}$ and $\{x', y', z'\}$ should be satisfy $z \neq z'$.

If these conditions are satisfied, then we get a well balanced family at least when we delete entirely a class of K_B and so we get in that case the values $b = b_{2p} - h(2t+1)$. But, if in the last class we delete only some blocks, we have to be careful in the choice of blocks to be deleted. That is easy for $h = 1$ or 2 and so by deleting another class of K_B we get a well balanced family for $b_{2p} - 3(2t+1) \leq b \leq b_{2p}$. But in the family for $b_{2p} - 3(2t+1)$ each element has been deleted exactly 3 times except z_1, z_2, u, where either $u = z_3$ (z_i appearing in the block $\{x_i, y_i, z_i\}$), if we apply deletion A-B 3 times or $u = \alpha$ if apply deletion A-B 2 timeswe and delete a complete class the third time. So if we want to have a well balanced family for $b_{2p} - (6t+4)$ we should delete the block $\{z_1, z_2, u\}$ which means it should not have been deleted before; but that is not always possible (see example after) when $p = 1$. Furthermore, if $p = 1$, then $h \leq 2t+1$. If $p > 1$, we can hope to get bigger values for h till $3t+1$. This method can give many possible values of b, but not all.

As example, for $n = 10$, take as K_A the KTS(9) of the Example 1 and as K_B the KTS(9) of Example 2. We can apply deletion A-B, by deleting $\{\alpha, \infty, \infty'\}$ and $\{3, 5, \alpha\}$ from the first class of K_A and the blocks of the two first classes of K_B except the blocks $\{1, \infty, \infty'\}, \{2, 3, 5\}$). We can no more apply the deletion with a pair in the first class of K_A, but if $p \geq 2$ we can delete $\{4, 6, \alpha\}$ keeping in K_B $\{4, 6, 0\}$. We delete the second and third class of K_B and finally the block $\{1, 2, 0\}$ of the fourth class getting a solution, when $p \geq 2$ for $b = 30p - 10$ where $\lambda_x = 9p - 3$ and $\lambda_{xy} = 2p$ or $2p - 1$. Deleting the 2 last blocks of the 4th class we get a well balanced family for $30p - 12 \leq b \leq 30p$, except for $b = 20$. But we will never get a solution for $b = 30p - 13$.

Another example can be given for $n = 16$. We take as K_A the Example 3 of $KTS(15)$ and modify by Construction A at least the first class and as K_B the developed of K_A by $+1$ (Example 4). In the first modified class of K_A , we can delete $\{0, 1, \alpha\}, \{2, 4, \alpha\}, \{7, 8, \alpha\}$. We keep in K_B, the blocks $\{1, 11, 0\}$, $\{2, 3, 4\}$, $\{7, 8, 10\}$ and delete the other blocks of the 5th, 6th and 7th classes of K_B. then we can delete the block $\{z_1, z_2, z_3\} = \{11, 3, 10\}$

getting a solution for all $80p - 16 \leq b \leq 80p$. If $p = 1$ we can continue the process with the deletion in the first class of K_A of $\{5, 10, \alpha\}, \{3, 6, \alpha\}$ (keeping in K_B in the 3rd class $\{5, 10, \infty'\}\{3, 6, \infty\}$). So we can get the values $80p - 31 \leq b \leq 80p$.

Addition process: We can also add to the well balanced family obtained for $b_{2p} = 2p(3t + 2)(2t + 1)$ some blocks of a KSTS(9) called K_B not used in the construction of Proposition 11. We can first add some blocks of a first parallel class. The family obtained is still well balanced, but when have added the whole class λ_α is one less than the other λ_x. So we have a well balanced family for $b_{2p} \leq b \leq b_{2p} + 2t + 1$.

We will apply the following construction called Construction B:

Construction B: Choose a class C of K_B, replace a block $\{x, y, z\}$ by the block $\{x, y, \alpha\}$ and add some of the other $2t$ blocks of this class. That works as soon as $\{x, y\}$ is not a pair appearing in a modified block of K_A (otherwise the block $\{x, y, \alpha\}$ will be repeated).

Doing construction B for one class C_1 of K_B is always possible as we have the freedom to chooses the classes to modify in K_A (or also to do the construction in K_B). We choose $\{x_1, y_1, z_1\}$ to be modify and the class of K_A containing $\{x_1, y_1\}$ will be not modified in construction A. So, adding some blocks of another class of K_B, we get a well balanced family for $b_{2p} + 2t + 2 \leq b \leq b_{2p} + 4t + 2$ (we have to add first the modified block containing α).

Then we can do construction B for another class C_2 of K_B. Having chosen $\{x_1, y_1\}$ in a class of K_A, we consider in the same class the triple containing z_1 let $\{z_1, x_2, a_2\}$ be this triple. We choose in K_B the class C_2 containing the pair $\{z_1, x_2\}$; let $\{z_1, x_2, z_2\}$ be the triple containing the pair $\{z_1, x_2\}$; we do construction B for this class replacing $\{z_1, x_2, z_2\}$ with $\{z_1, x_2, \alpha\}$ (note we are sure that $\{z_1, x_2\}$ is not in a modified class of K_A as it is in the same class as $\{x_1, y_1\}$). So we get a well balanced family for $b_{2p} + 4t + 3 \leq b \leq b_{2p} + 6t + 3$ (we have to add first the modified block $\{z_1, x_2, \alpha\}$.

At that point we have to be careful as $\lambda_{z_1}, \lambda_{z_2}, \lambda_\alpha$ are one less than the other λ_x and therefore we have to use a small trick. We choose as third block to be modified a block $\{z_2, x_3, z_3\}$, with $z_3 = x_1$ or y_1 and such that the block $\{z_1, z_2, z_3\}$ appears in a STS K_C different from K_A (it is different from K_B as $\{z_1, z_2\}$ appears in K_B with x_2). In order the construction works we need that the pair $\{z_2, x_3\}$ appears in a non modified class of K_A; that is always possible when $p < 3t$ and if we are lucky $\{z_2, x_3\}$appears in the same class of K_A as $\{x_1, y_1\}$ and $\{z_1, x_2\}$. Note that $\{z_2, x_3, z_3\}$ appears in a class C_3 of K_B different from C_1 (resp. C_2) as z_3 appears in $\{x_1, y_1, z_1\}$ (resp. Z_2 in $\{z_1, x_2, z_2\}$). Therefore, adding the 3 modified classes plus the block $\{z_1, z_2, z_3\}$ we get a well balanced family for $b = b_{2p} + 6t + 4$. Indeed all the λ_x have been increased by exactly 3 and some λ_{xy} by 1. We can then continue the process easily by adding the blocks of a 4th class of K_B, obtaining a well balanced family for $b_{2p} + 6t + 4 \leq b \leq b_{2p} + 8t + 5$. If $t = 1$ ($n = 10$), we cannot go further as we have used the 4 classes of K_B, but we know by proposition 6 that there does not exist a family for $b_{2p}+14$. If $t > 1$, we can continue the process by applying construction B for other classes of K_B. We have only to insure that the pair $\{x, y\}$ of the block $\{x, y, \alpha\}$

is not a pair appearing in a modified block of K_A and that we do the addition with disjoint pairs $\{x, y\}$ otherwise we will have created two pairs $\{x, \alpha\}$. Next examples show how to apply the additions for $n = 10$ and $n = 16$.

Case $n = 10$: Let K_A be the STS9 of Example 1 and K_B the STS(9) of Example 2. We do construction B with the first class of K_B, modifying the block $\{4, 6, 0\}$ of the first class to $\{\alpha, 6, 0\}$ and keeping $\{1, \infty, \infty'\}$ and $\{2, 3, 5\}$. Note that the pair $\{0, 6\}$ appears in the block $\{0, 1, 6\}$ of the 4th class of K_A. We have $z_1 = 4$, which appears in the block $\{4, 5, \infty'\}$ of the 4th class of K_A; here $x_2 = \infty'$. So we choose as C_2 the second class of K_B replacing $\{2, 4, \infty'\}$ by $\{\alpha, 4, \infty'\}$ and keeping the two other blocks $\{1, 3, 6\}$ and $\{5, 0, \infty\}$; here $z_2 = 2$. Then we choose in C_3 the block $\{2, 6, \infty\}$ (here $z_3 = y_1 = 6$) and we replace it by $\{2, \alpha, \infty\}$ keeping the two blocks $\{1, 4, 5\}$ and $\{3, 0, \infty'\}$. Note that we are lucky, as the pair $\{2, \infty\}$ is in the triple $\{2, 3, \infty\}$ which belongs also to the 4th class of K_A. We can now add the block $\{z_1, z_2, z_3\} = \{2, 4, 6\}$, which appears in the STS $K_C = K_A + 4$, different from K_A (it is obtained by adding 4 to $\{0, 2, 5\}$). We can then add blocks of the 4th class of K_B. In K_A we can modify by construction A any class except the 4th and so we can apply the construction for $p = 0, 1, 2, 3$ getting, as $b_{2p} = 30p$, the values $1 \leq b \leq 13$; $30 \leq b \leq 43$; $60 \leq b \leq 73$; $90 \leq b \leq 103$. Using Proposition 5 and the fact that $\binom{10}{3} = 120$ we get also the values $17 \leq b \leq 30$; $47 \leq b \leq 60$; $77 \leq b \leq 90$; $107 \leq b \leq 119$. So we get all the values except $b = 14, 15, 16, 44, 45, 46, 74, 75, 76, 104; 105, 106$; but we know by Proposition 6 that no well balanced family can exist for these values. So we have completely solve the case $n = 10$. In summary we have

Proposition 12. *For $n = 10$ conjecture 1 is verified; that is there exists a well balanced family for all b, except the $b \equiv 14, 15, 16$ (mod 30) for which such a family cannot exist.*

Case $n = 16$: We take as K_A the Example 3 of $KTS(15)$ and as K_B the Example 4. We first do construction B, by adding $\{8, 9, \alpha\}$, $\{4, 5, \alpha\}$, $\{\alpha, 0, \infty\}$ ($z_1 = \infty, z_2 = 9, z_3 = 4$) and the other blocks of the first, second and 4th class. Note that the pairs $\{4, 5\}, \{8, 9\}, \{0, \infty\}$ are in the 7th class of K_A. We can then add the block $\{z_1, z_2, z_3\} = \{9, \infty, 4\}$ which appears in the KTS $K_A + 3$ and also the blocks of another class. Doing so we get all the $b_{2p} = 80p \leq b \leq 80p + 21$. We can then replace $\{3, 10, 11\}$ by $\{3, 10, \alpha\}$ and $\{11, 6, 12\}$ by $\{11, 6, \alpha\}$. Note that the pairs $\{3, 10\}, \{11, 6\}$ are still in the 7th class of K_A. Adding the blocks of 2 other classes we get a well balanced family for all $80p \leq b \leq 80p + 31$ (we have to choose in K_A not to modify the 7th class in construction A). That is the best we can do for $p = 6$. If $p < 6$ we can add the block $\{7, 12, \alpha\}$ and all the other blocks of the 7th class of K_B. We have not to modify in K_A the class containing $\{7, 12\}$ namely the 3rd one (which is possible as we can leave 2 classes unmodified the 3rd and 7th). Then we can add the block $\{z_4, z_5, z_6\} = \{11, 12, 3\}$ which appears in $K_A + 8$ and finally blocks of the last class not used of K_B. Doing so we get a well balanced family for $80p \leq b \leq 80p + 37$ for $0 \leq p \leq 5$ and for $p = 6$, $480 \leq b \leq 511$. Using Proposition 5 and the fact that $\binom{16}{3} = 560$ we get also the values $49 \leq b \leq 80$; $80p - 37 \leq b \leq 80p$ for $2 \leq p \leq 7$ Note that for the values $80p + 38 \leq b \leq 80p + 42$, we know by Proposition 6 that no well balanced family can

exist. Therefore, to solve completely the case $n = 16$, it remains to deal with the values of b $43 \leq b \leq 48$. For that we will use a general construction called Construction C

Construction C: We take the blocks of an STS(n), $n \equiv 1$ or 3 (mod 6) S_C, plus $\frac{n+1}{2}$ blocks containing a new element α and $\frac{n+1}{2}$ pairs covering all the elements $0 \leq x \leq n-1$. let, for example these blocks be $\{2i, 2i+1, \alpha\}$ for $1 \leq i \leq \frac{n+1}{2}$ (modulo n). Doing so we get a well balanced family for $n+1$ and $b = \frac{n(n-1)}{6} + \frac{n+1}{2}$; indeed $\lambda_x = \frac{n+1}{2}$ except $\lambda_0 = \frac{n+1}{2} + 1$ and $\lambda_{xy} = 1$ except for the pairs $\{2i, 2i+1\}$ and $\{0, \alpha\}$ for which the value is 2. Then we can continue adding h disjoint blocks $1 \leq h \leq \frac{n}{3}$ not using 0 and not containing one of the pair for which the value is 2. We can continue the process as soon as we keep the balance. We will see later how it works for $n = 7$. When $n + 1 = 6t + 4$, we can also do construction C with construction A as soon as the pairs containing α are not in a modified class of K_A. We can also do that in a cleaver way starting from a KTS$(6t + 3)$ by taking $\frac{n+1}{2}$ pairs in a small number of classes (only 2 if possible). Then we can add the h blocks of a non used class replacing the block $\{x_0, y_0, z_0\}$ containing the x_0 which is repeated twice by the block $\{\alpha, y_0, z_0\}$. We get all the $(3t+1)(2t+1)+3t+2 \leq b \leq (3t+1)(2t+1)+5t+3 = (6t+4)(t+1)$. Note that we have for $b = (6t + 4)(t + 1)$ $\lambda_x = 3(t + 1)$ and $\lambda_{xy} = 1$ or 2. We can then continue adding a new class with a block modified and so on like we did in construction B. Let us show how Construction C works for $n = 16$.

We use the construction C by choosing as first STS(15) the K_A of Example 3 and picking the pairs in the KTS K_B of example 4. We add the triples $\{1, 10, \alpha\}$, $\{3, 0, \alpha\}$, $\{6, 12, \alpha\}$, $\{9, \infty, \alpha\}$, $\{4, \infty', \alpha\}$ obtained with pairs appearing in the first class of K_B. We also add $\{5, 11, \alpha\}$, $\{7, 8, \alpha\}$, $\{2, 4, \alpha\}$, with pairs appearing in the 5th class of K_B. We get a well balanced family for $b = 35 + 8 = 43$. Here 4 is repeated twice. Note that all these pairs appear in the 3 first classes of K_A; so we can also use Construction A for $0 \leq p \leq 4$. Then, we can add the blocks of the 3rd class replacing $\{1, 4, 12\}$ by $\{1, 12, \alpha\}$ ($\{1, 12\}$ appears also in the 3rd class of K_A) and we get the values $43 + 80p \leq b \leq 48 + 80p$, for $0 \leq p \leq 4$, and in particular the values $43 \leq b \leq 48$ which were the values still undecided for $n = 16$. Furthermore, we can add the blocks of the 2nd, 7th and 3rd classes of K_B, replacing $\{1, 3, 8\}$ by $\{3, 8, \alpha\}$, $\{2, 5, 6\}$ by $\{2, 5, \alpha\}$, and $\{7, 9, 11\}$ by $\{7, 11, \alpha\}$. Then we can add the block $\{1, 6, 9\}$ in the 5th class. We finally add blocks of the 4th class and the block $\{12, 0, \alpha\}$ of the 5th class. Doing so we have used also pairs in the 5th and 6th class of K_A. We got all the values $43 + 80p \leq b \leq 70 + 80p$ for $p = 0, 1, 2$.

To summarize using all the constructions we were able to prove Conjecture 1 for $n = 16$.

Proposition 13. *For $n = 16$, Conjecture 1 is verified; that is there exists a well balanced family for all b except the $b \equiv 38, 39, 40, 41, 42$ (mod 80) for which such a family cannot exist.*

Small cases

n=5

For $n = 5$, $\binom{5}{3} = 10$ and by Proposition 5 we have to consider only the values of $b \leq 5$.

We have well balanced families for $b = 1$ (one block) and $b = 2$ (two blocks $\{1, 2, 3\}$ and $\{1, 4, 5\}$), but not for $b = 3$ as we have seen in the example of the introduction (see also Proposition 7). However there exists an optimal solution $\{1, 2, 3\}, \{1, 2, 4\}, \{3, 4, 5\}$ 1-balanced but not 2-balanced ($\lambda_{12} = 2$ but $\lambda_{15} = \lambda_{25} = 0$). By Proposition 7, there is no well balanced solution for $b = 4$; an optimal one consists of the blocks $\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 5\}, \{3, 4, 5\}$. For $b = 5$ there exists a well balanced solution with $\lambda_x = 3$ and $\lambda_{xy} = 1$ or 2 and consisting of the 5 blocks $\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 5\}, \{3, 4, 5\}, \{2, 4, 5\}$.

n=6

For $n = 6$, $\binom{6}{3} = 20$ and by Proposition 5 we have to consider only the values of $b \leq 10$.

For $b = 5$ (and so $b = 15$), there does not exist by Proposition 6 a well balanced family. An optimal solution $\mathcal{F}^*$ consists of the 5 blocks:
$\{1, 2, 3\}, \{1, 2, 4\}, \{1, 5, 6\}, \{2, 5, 6\}, \{3, 4, 5\}$ ($\lambda_x = 2$ or 3 and $\lambda_{12} = \lambda_{56} = 2$ but $\lambda_{36} = \lambda_{46} = 0$) with $4x^2 + 16x$ as associated polynomial. The proof is obtained by looking at various cases. Let a general solution be of the form $P(\mathcal{F}, x) = \alpha x^3 + \beta x^2 + \gamma x + \delta$. Furthermore, we always have $\alpha + \beta + \gamma = b(b - 1) = 20$. So $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) = (x - 1)(\alpha(x - 1)^2 + (3\alpha + \beta - 4)(x - 1) + 2\alpha + \beta - 4 - \delta)$. If $\alpha \geq 2$ (that is at least one block repeated), one can show by looking at the number of repeated blocks that $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) > 0$ for any $x > 1$. For example suppose one block is repeated 3 times say $\{1, 2, 3\}$; then either we have twice the block $\{4, 5, 6\}$ and then $\alpha = 8$ and $\delta = 12$ otherwise $\alpha = 6$ and $\delta \leq 6$.

In what follows, let $\alpha = 0$, that is no repeated block. Then $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) = (\beta - 4)(x - 1) + \beta - 4 - \delta)$. Note that $\delta \leq 4$, as there can be at most two pairs of disjoint blocks (as $b = 5$); furthermore, when we have two disjoint blocks any other block intersect one block in 2 elements and the other in one element. We distinguish 3 cases :

- $\delta = 4$. Wlog let the pairs of disjoint blocks be $B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}, B_3 = \{1, 2, 4\}, B_4 = \{3, 5, 6\}$. Then B_1 and B_3 intersect in a pair and also B_2 and B_4; the last block B_5 intersect one of B_1, B_2 and one of B_3, B_4 in a pair; so $\beta = 8$ and $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) \geq 0$ for $x \geq 1$.
- $\delta = 2$. Let the two disjoint blocks be B_1 and B_2. Then any other block intersect one of them in a pair and so $\beta \geq 6$ and $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) \geq 0$ for $x \geq 1$.
- $\delta = 0$. First, $\beta = 0$ is impossible as if $B_1 = \{1, 2, 3\}$, we have at most 3 pairs available among $\{4, 5, 6\}$. If $\beta = 2$; let the two blocks intersecting in a pair be $B_1 = \{1, 2, 3\}, B_2 = \{1, 2, 4\}$; the pair $\{5, 6\}$ can be in at most one triple, but the two other blocks need to share either one pair with B_1 or B_2 or the pair $\{3, 4\}$. So necessarily $\beta \geq 4$ and $P(\mathcal{F}, x) - P(\mathcal{F}^*, x) \geq 0$ for $x \geq 1$.

For the other values of b, we can construct well balanced families as follows. Let $B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}$; $C_1 = \{1, 2, 4\}, C_2 = \{1, 3, 5\}, C_3 = \{2, 3, 6\}$; $D_1 = \{1, 4, 6\}, D_2 = \{2, 5, 6\}, D_3 = \{3, 4, 5\}$ and $C'_1 = \{1, 2, 5\}, C'_2 = \{1, 3, 6\}, C'_3 = \{2, 3, 4\}$. Note that the C_i and C'_i (resp D_i) intersect B_1 (resp. B_2) in three different pairs and B_2 (resp B_1) in 3 different elements. Solutions are obtained by taking:

- for $b = 1$ B_1;
- for $b = 2$, B_1, B_2;
- for $b = 3$, C_1, C_2, C_3;
- for $b = 4$, C_1, C_2, C_3, B_2;
- for $b = 6$, $C_1, C_2, C_3, D_1, D_2, D_3$;
- for $b = 7$, $C_1, C_2, C_3, D_1, D_2, D_3, B_1$;
- for $b = 8$, $C_1, C_2, C_3, D_1, D_2, D_3, B_1, B_2$;
- for $b = 9$, $C_1, C_2, C_3, D_1, D_2, D_3, C'_1, C'_2, C'_3$;
- for $b = 10$, $C_1, C_2, C_3, D_1, D_2, D_3, C'_1, C'_2, C'_3, B_2$.

n=7

For $n = 7$, $\binom{7}{3} = 35$; by Proposition 3 and Proposition 5 we have to consider only the values of $b \leq 17$.

Proposition 14. *For $k = 3$ and $n = 7$, there exists a balanced family for any b.*

Proof. Kirkman proved that there exist two disjoint STS(7). The first one consists of the 7 blocks $C_i = \{i, i+1, i+3\}$, for $0 \leq i \leq 6$ and the second one of the 7 blocks $D_i = \{i, i+2, i+3\}$, for $0 \leq i \leq 6$ (indices modulo 7). Let $B_1 = \{1, 2, 3\}, B_2 = \{4, 5, 6\}, B_3 = \{0, 1, 4\}, B_4 = \{0, 2, 5\}, B_5 = \{0, 3, 6\}$. For $b = j, 1 \leq j \leq 5$ take the blocks $B_i, 1 \leq i \leq j$. For $b = 7$ take the first STS(7) (that is all the C_i). For $b = 6$ delete one block from the STS(7). For $b = 7 + j, 1 \leq j \leq 5$ add to the STS(7) the blocks $B_i, 1 \leq i \leq j$. For $b = 14$ take the two disjoint STS(7) (that is all the C_i and D_i). For $b = 13$ delete one block from one STS(7). For $b = 14 + j, 1 \leq j \leq 5$ add to the two disjoint STS(7) the blocks $B_i, 1 \leq i \leq j$. □

n=8 For $n = 8$, $\binom{8}{3} = 56$ and by Proposition 5 we have to consider only the values of $b \leq 28$. By Proposition 6 and 7 there do not exist well balanced families for $b = 9, 10, 18, 19, 27, 28$. For the other values let us construct a well balanced family. By proposition 10 we have a solution for $b = 8$, consiting of the 8 blocks obtained by deleting ∞' in the Example 1 of a KTS(9) namely (we relabel ∞ with 7).

$$\begin{array}{llll} B_1 = \{1, 2, 4\} & B_3 = \{0, 2, 5\} & B_5 = \{0, 3, 4\} & B_7 = \{0, 1, 6\} \\ B_2 = \{3, 5, 6\} & B_4 = \{4, 6, 7\} & B_6 = \{1, 5, 7\} & B_8 = \{2, 3, 7\} \end{array}$$

For $b = 2q$ we have a well balanced family by taking the blocks $B_j, 1 \leq j \leq 2q$; For $b = 3$ (resp; $b = 5$) add to B_1, B_2 (resp B_1, B_2, B_3, B_4) the block $\{0, 1, 7\}$. For $b = 7$ take the blocks $B_j, 1 \leq j \leq 7$.

We can also use the STS(9) to construct solutions for $b = 12$ and other values; but it is better to use Construction C starting from the STS(7) with the 7 blocks $C_i = \{i, i+1, i+3\}$, for $0 \leq i \leq 6$ (values modulo 7) and to add a new element α. For $b = 11$, we add to the STS(7) the 4 blocks $E_1 = \{0, 1, \alpha\}$, $E_2 = \{2, 3, \alpha\}$, $E_3 = \{4, 5, \alpha\}$, $E_4 = \{0, 6, \alpha\}$. Note that $\lambda_x = 4$ except $\lambda_0 = 5$ and $\lambda_{xy} = 1$ except for the pairs $\{0, 1\}$, $\{2, 3\}$, $\{4, 5\}$, $\{0, 6\}$ and $\{0, \alpha\}$. Then we can add sucessively $E_5 = \{1, 3, 5\}$, $E_6 = \{2, 4, 6\}$, $E_7 = \{1, 2, \alpha\}$, $E_8 = \{0, 3, 4\}$, $E_9 = \{5, 6, \alpha\}$, and $E_{10} = \{0, 2, 5\}$ getting solutions for $11 \leq b \leq 17$. For $b = 20$ we add furthermore the 3 blocks $E_{11} = \{1, 4, 6\}$, $E_{12} = \{3, 6, \alpha\}$, $E_{13} = \{0, 4, \alpha\}$. One can note that all these blcocks are disjoint from those of teh STS.

For $21 \leq b \leq 26$, we will use construction C, starting with the two disjoint STS(7) with blocks $C_i = \{i, i+1, i+3\}$, for $0 \leq i \leq 6$ and the second one with blocks $D_i = \{i, i+2, i+3\}$, for $0 \leq i \leq 6$. (indices modulo 7). Add the 7 blocks $F_i = \{i, i + 1, \alpha\}$, $0 \leq i \leq 6$. We get a solution for $b = 21$. Note that $\lambda_x = 8$ except $\lambda_\alpha = 7$ and $\lambda_{xy} = 2$ except for the pairs $\{i, i+1\}$. Then add the blocks $\{0, 4, \alpha\}$, $\{2, 6, \alpha\}$, $\{1, 3, 5\}$ (at that point for $b = 24$, $\lambda_x = 9$) and $\{0, 2, 5\}$, $\{1, 4, 6\}$.

7 Case $k > 3$

References

[BBJMR08] A-E. Baert, V. Boudet, A. Jean-Marie, and X. Roche. Minimization of download times in a distributed vod system. In *ICPP08: The international conference on parallel processing*, pages 173–180, Los Alamitos, CA, USA, 2008. IEEE.

[BBJMR09a] A-E. Baert, V. Boudet, A. Jean-Marie, and X. Roche. Minimization of download time variance in a distributed vod system. *Scalable Computing Practice and experience*, 10(1):75–86, 2009.

[BBJMR09b] A-E. Baert, V. Boudet, A. Jean-Marie, and X. Roche. Performance analysis of dat replication in grid delivery networks. In *Int. Conf. on Complex Intelligent and Software Intensive Systems*, pages 369–374, 2009.

[CJ06] Colbourn C.J. and Dinitz J.H., editors. *The CRC Handbook of Combinatorial Designs (2nd edition)*, volume 42. CRC Press, 2006.

[CM06] C. J. Colbourn and R. Mathon. *The CRC Handbook of Combinatorial Designs (2nd edition)*, volume 42 of *Discrete Mathematics and Its Applications*, chapter *Steiner systems, pages 102–110. CRC Press, C.J. Colbourn and J.H. Dinitz edition, 2006.*

[Ji05] *L. Ji. A new existence proof for large sets of disjoint steiner triple systems.* J. Combinatorial theory A*, 112:308–327, 2005.*

[JMRBB09] *A. Jean-Marie, X. Roche, A-E. Baert, and V. Boudet. Combinatorial designs and availability. Technical Report RR 7119, INRIA, 2009.*

[Lu83] *J.X. Lu. On large sets of disjoint steiner triple systems i,ii, iii.* J. Combinatorial theory A*, 34:140–146,147–155, and 156–182, 1983.*

[Lu84] *J.X. Lu. On large sets of disjoint steiner triple systems iv,v, vi.* J. Combinatorial theory A*, 37:136–163,164–188, and 189–192, 1984.*

[RCW71] *D.K. Ray-Chauduri and R.M. Wilson. Solution of kirkman's schoolgirl problem.* Proc. symp. pure Math*, 19:187–204, 1971.*

[Tei91] *L. Teirlinck. A completion of lu's determination of the spectrum of large sets of disjoint steiner triple systems.* J. Combinatorial theory A*, 57:302–305, 1991.*

[ZC10] *J. Zhou and Y. Chang. new results on large sets of kirkman triple systems.* Des Codes Cryptogr*, 55:1–7, 2010.*

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399

Chapitre 6

Perspectives générales

Dans cette thèse, nous avons abordé quatre problèmes importants dans les réseaux (reconfiguration du routage dans les réseaux optiques, économie d'énergie dans les réseaux cœur, ordonnancement des liens dans les réseaux sans-fil, placement de données dans les réseaux pair-à-pair). Pour chacun d'entre eux, nous avons adopté une modélisation précise et relativement simplifiée de manière à pouvoir obtenir des résultats significatifs d'un point de vue théorique et pratique (heuristiques, simulations) même si la plupart des problèmes considérés sont NP-difficiles pour des instances très simples. Dans les conclusions de chaque chapitre, nous avons indiqué des problèmes précis qui restent ouverts. Ces études nous ont amenés à introduire et développer des paramètres de théorie des graphes et des configurations qui méritent une étude encore plus approfondie indépendamment de l'application réseaux. Il reste évidemment encore beaucoup de travail à faire sur chacun de ces problèmes avant de pouvoir transférer nos résultats dans les réseaux réels. De plus, pour chacun de ces problèmes, d'autres modélisations peuvent être envisagées qui appréhenderaient peut être mieux la réalité.

Plus précisément, pour le problème de reconfiguration du routage dans les réseaux optiques, nous avons obtenu de nombreux résultats de complexité ou d'algorithmique notamment. De nombreuses pistes doivent encore être développées comme la prise en compte des contraintes physiques réels dans ce genre de réseaux ou encore de la durée des interruptions. En effet, le nombre d'interruptions (simultané et/ou total) a été étudié au cours de cette thèse mais la durée des différentes interruptions apparaît comme étant un paramètre central pour ce problème. Cette étude mérite donc la poursuite des efforts pour concevoir des modèles pertinents et ainsi obtenir des résultats significatifs.

Le problème d'économie d'énergie dans les réseaux cœur est un problème récent et comme nous l'avons montré, éteindre des équipements (interfaces) permet d'économiser de l'énergie, au détriment parfois de la tolérance aux pannes et de la longueur des routes. Ces solutions ne sont pour le moment pas implémentées dans les réseaux existants et peuvent se révéler techniquement difficiles. De plus, le coût de remise en route de ces équipements est peut être loin d'être négligeable. Des mesures pratiques doivent être réalisées et d'autres manières d'attaquer le problème doivent être envisagées.

L'ordonnancement distribué des liens dans les réseaux sans-fil est un problème central dans ce type de réseaux. Outre les problèmes ouverts mis en exergue dans le chapitre correspondant, il est nécesaire de poursuivre ce travail notamment pour que

notre modèle représente encore mieux la réalité. Entre autres extensions, la prise en compte d'un trafic multi-hop sans routage prédéfini et la prise en compte de modèles d'interférence asymétriques apparaissent très importants pour ce problème.

Le placement optimal des données dans les réseaux pair-à-pair nous a amenés à considérer des configurations "bien équilibrées" : chaque t-uplet appartenant au même nombre (à un près) de blocs. L'existence de telles configurations amène une nouvelle problématique, plus générale que celle des "designs" classiques, qui apparaît très prometteuse et devrait donner lieu à de nombreux développements ultérieurs. La résolution du seul cas particulier $k = 3$ nécessitera vraisemblablement l'étude de nombreux cas et pourra occuper plusieurs chercheurs dans les années à venir.

Annexe A

Coloration impropre pondérée

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Weighted Improper Colouring

J. Araujo — J.-C. Bermond — F. Giroire — F. Havet — D. Mazauric — R. Modrzejewski

N° 7590

Octobre 2011

Domaine 3

ISSN 0249-6399 ISRN INRIA/RR--7590--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE | INRIA | centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Weighted Improper Colouring *

J. Araujo † ‡ , J.-C. Bermond † , F. Giroire † , F. Havet † , D. Mazauric † , R. Modrzejewski†

Thème : Réseaux, systèmes et services, calcul distribué
Équipe-Projet Mascotte

Rapport de recherche n° 7590 — Octobre 2011 — 23 pages

Abstract: In this paper, we study a colouring problem motivated by a practical frequency assignment problem and, up to our best knowledge, new. In wireless networks, a node interferes with other nodes, the level of interference depending on numerous parameters: distance between the nodes, geographical topography, obstacles, etc. We model this with a weighted graph (G,w) where the weight function w on the edges of G represents the noise (interference) between the two end-vertices. The total interference in a node is then the sum of all the noises of the nodes emitting on the same frequency. A weighted t-improper k-colouring of (G,w) is a k-colouring of the nodes of G (assignment of k frequencies) such that the interference at each node does not exceed some threshold t. We consider here the Weighted Improper Colouring problem which consists in determining the weighted t-improper chromatic number defined as the minimum integer k such that (G,w) admits a weighted t-improper k-colouring. We also consider the dual problem, denoted the Threshold Improper Colouring problem, where, given a number k of colours (frequencies), we want to determine the minimum real t such that (G,w) admits a weighted t-improper k-colouring. We show that both problems are NP-hard and first present general upper bounds; in particular we show a generalisation of Lovász's Theorem for the weighted t-improper chromatic number. We then show how to transform an instance of the Threshold Improper Colouring problem into another equivalent one where the weights are either 1 or M, for a sufficient large M. Motivated by the original application, we study a special interference model on various grids (square, triangular, hexagonal) where a node produces a noise of intensity 1 for its neighbours and a noise of intensity 1/2 for the nodes at distance 2. Consequently, the problem consists in determining the weighted t-improper chromatic number when G is the square of a grid and the weights of the edges are 1 if their end-vertices are adjacent in the grid, and 1/2 if their end-vertices are linked in the square of the grid, but not in the grid. Finally, we model the problem using integer linear programming, propose and test heuristic and exact Branch-and-Bound algorithms on random cell-like graphs, namely the Poisson-Voronoi tessellations.

Key-words: graph colouring, improper colouring, interference, radio networks, frequency assignment, grids, algorithms

* This work was partially supported by région PACA, ANR Blanc AGAPE (ANR-09-BLAN-0159) and ANR International Taiwan GRATEL (ANR-09-BLAN-0373).

† {julio.araujo, jean-claude.bermond, frederic.giroire, frederic.havet, dorian.mazauric, remigiusz.modrzejewski}@inria.fr - MASCOTTE Project, I3S (CNRS & UNS) and INRIA, INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex France.

‡ ParGO Research Group - Universidade Federal do Ceará - UFC - Campus do Pici, Bloco 910. 60455-760 - Fortaleza, CE - Brazil.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Coloration Impropre Pondérée

Résumé : Dans ce papier, nous étudions un nouveau problème de coloration motivé par un problème pratique d'allocation de fréquences. Dans les réseaux sans-fil, un nœud interfère avec d'autres, à un niveau dépendant de nombreux paramètres : la distance entre les nœuds, la topographie physique, les obstacles, etc. Nous modélisons cela par un graphe arête-valué (G,w) où le poids d'une arête représente le bruit (ou l'interférence) entre ces deux extrémités. L'interférence totale au niveau d'un nœud est alors la somme de tous les bruits entre ce nœud et les autres nœuds émettant avec la même fréquence. Une k-coloration t-impropre pondérée de (G,w) est une k-coloration des nœuds de G (assignation de k fréquences) telle que l'interférence à chaque nœud n'excède pas un certain seuil t. Dans ce papier, nous étudions le problème de la coloration impropre pondérée qui consiste à déterminer le nombre chromatique t-impropre pondéré d'un graphe arête-valué (G,w), qui est le plus petit entier k tel que (G,w) admette une k-coloration t-impropre pondérée. Nous considérons également le problème Threshold Improper Colouring (le problème dual) qui, étant donné un nombre de couleurs (fréquences), consiste à déterminer le plus petit réel t tel que (G,w) admette une k-coloration t-impropre pondérée. Nous montrons que tous ces problèmes sont NP-difficiles en présentant tout d'abord des bornes supérieures générales; en particulier nous prouvons une généralisation du théorème de Lovász pour le problème du nombre chromatique t-impropre pondéré. Nous montrons ensuite comment transformer une instance du problème Threshold Improper Colouring en une instance équivalente avec des poids 1 ou M, pour une valeur de M suffisamment grande. Motivé par l'origine du problème, nous étudions un modèle d'interférence particulier pour différentes grilles (carrée, triangulaire, hexagonale) où un nœud produit un bruit d'intensité 1 pour ses voisins et un bruit d'intensité $1/2$ pour les nœuds à distance 2. Le problème consiste alors à déterminer le nombre chromatique t-impropre pondéré lorsque G est le carré d'une grille et que les poids des arêtes valent 1 si les deux nœuds extrémités sont adjacents dans la grille, et $1/2$ sinon. Enfin, nous modélisons le problème par des programmes linéaires en nombres entiers, nous proposons des heuristiques et des algorithmes exactes d'énumération par la technique du Branch-and-Bound. Nous les testons pour des graphes aléatoires ressemblant à des réseaux cellulaires, à savoir des tessellations de Poisson-Voronoi.

Mots-clés : coloration de graphes, coloration impropre, interférence, allocation de fréquences, réseaux radio, grilles, algorithmes

1 Introduction

Let $G = (V,E)$ be a graph. A *k-colouring* of G is a function $c : V \to \{1,\ldots,k\}$. The colouring c is *proper* if $uv \in E$ implies $c(u) \neq c(v)$. The *chromatic number* of G, denoted by $\chi(G)$, is the minimum integer k such that G admits a proper k-colouring. The goal of the VERTEX COLOURING problem is to determine $\chi(G)$ for a given graph G. It is a well-known NP-hard problem [12].

A k-colouring c is *l-improper* if $|\{v \in N(u) \mid c(v) = c(u)\}| \leq l$ for all $u \in V$. Given a non-negative integer l, the *l-improper chromatic number* of a graph G, denoted by $\chi_l(G)$, is the minimum integer k such that G admits an l-improper k-colouring. Given a graph G and an integer l, the IMPROPER COLOURING problem consists in determining $\chi_l(G)$ and is also NP-hard [5, 15]. Indeed, if $l = 0$, observe that $\chi_0(G) = \chi(G)$. Consequently, VERTEX COLOURING is a particular case of IMPROPER COLOURING.

In this work we define and study a new variation of the IMPROPER COLOURING problem for edge-weighted graphs. An edge-weighted graph is a pair (G,w) where $G = (V,E)$ is a graph and $w : E \to \mathbb{R}^*_+$. Given an edge-weighted graph (G,w) and a colouring c of G, the *interference* of a vertex u in this colouring is defined by

$$I_u(G,w,c) = \sum_{\{v \in N(u) | c(v) = c(u)\}} w(u,v).$$

For any non-negative real number t, called *threshold*, we say that c is a *weighted t-improper k-colouring* of (G,w) if c is a k-colouring of G such that $I_u(G,w,c) \leq t$, for all $u \in V$.

Given a threshold $t \in \mathbb{R}^*_+$, the minimum integer k such that the graph G admits a weighted t-improper k-colouring is the *weighted t-improper chromatic number* of (G,w), denoted by $\chi_t(G,w)$. Given an edge-weighted graph (G,w) and a threshold $t \in \mathbb{R}^*_+$, determining $\chi_t(G,w)$ is the goal of the WEIGHTED IMPROPER COLOURING problem. Note that if $t = 0$ then $\chi_0(G,w) = \chi(G)$, and if $w(e) = 1$ for all $e \in E$, then $\chi_l(G,w) = \chi_l(G)$ for any positive integer l. Therefore, the WEIGHTED IMPROPER COLOURING problem is clearly NP-hard since it generalises VERTEX COLOURING and IMPROPER COLOURING.

On the other hand, given a positive integer k, we define the *minimum k-threshold* of (G,w), denoted by $T_k(G,w)$ as the minimum real t such that (G,w) admits a weighted t-improper k-colouring. Then, for a given edge-weighted graph (G,w) and a positive integer k, the THRESHOLD IMPROPER COLOURING problem consists of determining $T_k(G,w)$. The THRESHOLD IMPROPER COLOURING problem is also *NP*-hard. This fact follows from the observation that determining whether $\chi_l(G) \leq k$ is NP-complete, for every $l \geq 2$ and $k \geq 1$ [5–7]. Consequently, in particular, it is a *NP*-complete problem to decide whether a graph G admits a weighted t-improper 2-colouring when all the weights of the edges of G are equal to one, for every $t \geq 2$.

1.1 Motivation

Our initial motivation to these problems was the design of satellite antennas for multi-spot MFTDMA satellites [2]. In this technology, satellites transmit signals to areas on the ground called *spots*. These spots form a grid-like structure which is modelled by an hexagonal cell graph. To each spot is assigned a radio channel or colour. Spots are interfering with other spots having the same channel and a spot can use a colour only if the interference level does not exceed a given threshold t. The level of interference between two spots depends on their distance. The authors of [2] introduced a factor of mitigation γ and the interferences of remote spots are reduced by a factor $1-\gamma$. When the interference level is too low, the nodes are considered to not interfere anymore. Considering such types of interferences, where nodes at distance at most i interfere, leads to the study of the i-th power of the graph modelling the network and a case of special interest is the power of grid graphs (see Section 3).

1.2 Related Work

Our problems are particular cases of the FREQUENCY ASSIGNMENT problem (FAP). FAP has several variations that were already studied in the literature (see [1] for a survey). In most of these variations, the main constraint to be satisfied is that if two vertices (mobile phones, antennas, spots, etc.) are close, then the difference between the frequencies that are assigned to them must be greater than some function which usually depends on their distance.

There is a strong relationship between most of these variations and the $L(p_1,\dots,p_d)$-LABELLING problem [16]. In this problem, the goal is to find a colouring of the vertices of a given graph G, in such a way that the difference between the colours assigned to vertices at distance i is at least p_i, for every $i = 1,\dots,d$.

In some other variants, for each non-satisfied interference constraint a penalty must be paid. In particular, the goal of the MINIMUM INTERFERENCE ASSIGNMENT problem (MI-FAP) is to minimise the total penalties that must be paid, when the number of frequencies to be assigned is given. This problem can also be studied for only *co-channel interferences*, in which the penalties are applied only if the two vertices have the same frequency. However, MI-FAP under these constraints does not correspond to WEIGHTED IMPROPER COLOURING, because we consider the co-channel interference, i.e. penalties, just between each vertex and its neighbourhood.

The two closest related works we found in the literature are [14] and [8]. However, they both apply penalties over co-channel interference, but also to the *adjacent channel interference*, i.e. when the colours of adjacent vertices differ by one unit. Moreover, their results are not similar to ours. In [14], they propose an enumerative algorithm for the problem, while in [8] a Branch-and-Cut method is proposed and applied over some instances.

1.3 Results

In this article, we study both parameters $\chi_t(G,w)$ and $T_k(G,w)$. We first present general upper bounds; in particular we show a generalisation of Lovász's Theorem for $\chi_t(G,w)$. We after show how to transform an instance of THRESHOLD IMPROPER COLOURING into an equivalent one where the weights are either 1 or M, for a sufficient large M.

Motivated by the original application, we study a special interference model on various grids (square, triangular, hexagonal) where a node produces a noise of intensity 1 for its neighbours and a noise of intensity 1/2 for the nodes that are at distance 2. Consequently, the problem consists in determining $\chi_t(G,w)$ and $T_k(G,w)$, when G is the square of a grid and the weights of the edges are 1, if their end-vertices are adjacent in the grid, and 1/2 if they are adjacent in the square of the grid, but not in the grid.

Finally, we propose a heuristic and a Branch-and-Bound algorithm to solve THRESHOLD IMPROPER COLOURING for general graphs. We compare them to an integer linear programming formulation on random cell-like graphs, namely Voronoi diagrams of random points of the plan. These graphs are classically used in the literature to model telecommunication networks [3, 9, 10].

2 General Results

In this section, we present some results for WEIGHTED IMPROPER COLOURING and THRESHOLD IMPROPER COLOURING for general graphs and general interference models.

2.1 Upper bounds

Let (G,w) be an edge-weighted graph with $w : E(G) \to \mathbb{R}_+^*$. For any vertex $v \in V(G)$, its *weighted degree* is $d_w(v) = \sum_{u\in N(v)} w(u,v)$. The *maximum weighted degree* of G is $\Delta(G,w) = \max_{v\in V} d_w(v)$.

Given a k-colouring $c : V \to \{1,\dots,k\}$ of G, we define $d^i_{w,c}(v) = \sum_{\{u \in N(v) | c(u)=i\}} w(u,v)$, for every vertex $v \in V(G)$ and colour $i = 1,\dots,k$. Note that $d^{c(v)}_{w,c}(v) = I_v(G,w,c)$. We say that a k-colouring c of G is *w-balanced* if c satisfies the following property:

$$\text{For any vertex } v \in V(G),\ I_v(G,w,c) \leq d^j_{w,c}(v),\ \text{ for every } j = 1,\dots,k.$$

We denote by $\gcd(w)$ the greatest common divisor of the weights of w. We use here the generalisation of the gcd to non-integer numbers (e.g. in $\mathbb{Q}$) where a number x is said to divide a number y if the fraction y/x is an integer. The important property of $\gcd(w)$ is that the difference between two interferences is a multiple of $\gcd(w)$; in particular, if for two vertices v and u, $d^i_{w,c}(v) > d^j_{w,c}(u)$, then $d^i_{w,c}(v) \geq d^j_{w,c}(u) + \gcd(w)$.

If t is not a multiple of the $\gcd(w)$, that is, there exists an integer $a \in \mathbb{Z}$ such that $a\ \gcd(w) < t < (a+1)\gcd(w)$, then $\chi^w_t(G) = \chi^w_{a\ \gcd(w)}(G)$.

Proposition 1. *Let (G,w) be an edge-weighted graph. For any $k \geq 2$, there exists a w-balanced k-colouring of G.*

Proof. Let us colour $G = (V,E)$ arbitrarily with k colours and then repeat the following procedure: if there exists a vertex v coloured i and a colour j such that $d^i_{w,c}(v) > d^j_{w,c}(v)$, then recolour v with colour j. Observe that this procedure neither increases (we just move a vertex from one colour to another) nor decreases (a vertex without neighbour on its colour is never moved) the number of colours within this process. Let W be the sum of the weights of the edges having the same colour in their end-vertices. In this transformation, W has increased by $d^j_{w,c}(v)$ (edges that previously had colours i and j in their end-vertices), but decreased by $d^i_{w,c}(v)$ (edges that previously had colour i in both of their end-vertices). So, W has decreased by $d^i_{w,c}(v) - d^j_{w,c}(v) \geq \gcd(w)$. As $W \leq |E| \max_{e \in E} w(e)$ is finite, this procedure finishes and produces a w-balanced k-colouring of G. □

The existence of a w-balanced colouring gives easily some upper bounds on the weighted t-improper chromatic number and the minimum k-threshold of an edge-weigthed graph (G,w). It is a folklore result that $\chi(G) \leq \Delta(G)+1$, for any graph G. Lovász [13] extended this result for IMPROPER COLOURING problem using w-balanced colouring. He proved that $\chi_l(G) \leq \lceil \frac{\Delta(G)+1}{l+1} \rceil$. In what follows, we extend this result to weighted improper colouring.

Theorem 2. *Let (G,w) be an edge-weighted graph with $w : E(G) \to \mathbb{Q}^*_+$, and t a multiple of* $\gcd(w)$. *Then*

$$\chi_t(G,w) \leq \left\lceil \frac{\Delta(G,w) + \gcd(w)}{t + \gcd(w)} \right\rceil.$$

Proof. If $k = 1$ there is nothing to prove.

Observe that, in any w-balanced k-colouring c of a graph G, the following holds:

$$d_w(v) = \sum_{u \in N(v)} w(u,v) \geq k d^{c(v)}_{w,c}(v). \tag{1}$$

Let $k^* = \left\lceil \frac{\Delta(G,w)+\gcd(w)}{t+\gcd(w)} \right\rceil \geq 2$ and c^* be a w-balanced k^*-colouring of G. We claim that c^* is a weighted t-improper k^*-colouring of (G,w).

By contradiction, suppose that there is a vertex v in G such that $c^*(v) = i$ and that $d^i_{w,c}(v) > t$. Since c^* is w-balanced, $d^j_{w,c}(v) > t$, for all $j = 1,\dots,k^*$. By the definition of $\gcd(w)$ and as t is a multiple of $\gcd(w)$, it leads to $d^j_{w,c}(v) \geq t + \gcd(w)$ for all $j = 1,\dots,k^*$. Combining this inequality with Inequality (1), we obtain:

$$\Delta(G,w) \geq d_w(v) \geq k^*(t + \gcd(w)),$$

giving

$$\Delta(G,w) \geq \Delta(G,w) + \gcd(w),$$

a contradiction. The result follows.

□

Note that when all weights are unit, we obtain the bound for the improper colouring derived in [13]. Brooks [4] proved that for a connected graph G, $\chi(G) = \Delta(G) + 1$ if, and only if, G is complete or an odd cycle. One could wonder for which edge-weighted graphs the bound we provided in Theorem 2 is tight. However, Correa *et al.* [5] already showed that it is NP-complete to determine if the improper chromatic number of a graph G attains the upper bound of Lovász, which is a particular case of WEIGHTED IMPROPER COLOURING , i.e., of the bound Theorem 2.

The w-balanced colourings also yields upper bounds for the minimum k-threshold of an edge-weighted graph (G,w). When $k = 1$, then all the vertices must be coloured same, and $T_1(G,w) = \Delta(G,w)$. This may be generalized as follows, using w-balanced colourings.

Theorem 3. *Let (G,w) be an edge-weighted graph with $w : E(G) \to \mathbb{R}_+$, and let k be a positive integer. Then*

$$T_k(G,w) \leq \frac{\Delta(G,w)}{k}.$$

Proof. Let c be a w-balanced k-colouring of G. Then

$$kT_k(G,w) \leq kd_{w,c}^{c(v)}(v) \leq d_w(v) = \sum_{u \in N(v)} w(u,v) \leq \Delta(G,w)$$

□

Because $T_1(G,w) = \Delta(G,w)$, Theorem 3 may be restated as $T_k(G,w) \leq T_1(G,w)$. This inequality may be generalised as follows.

Theorem 4. *Let (G,w) be an edge-weighted graph with $w : E(G) \to \mathbb{R}_+$, and let k and p be two positive integers. Then*

$$T_{kp}(G,w) \leq \frac{T_p(G,w)}{k}.$$

Proof. Set $t = T_p(G,w)$. Let c be a t-improper p-colouring of (G,w). For $i = 1,\ldots,p$, let G_i be the subgraph of G induced by the vertices coloured i by c. By definition of improper colouring $\Delta(G_i,w) \leq t$ for all $1 \leq i \leq p$. By Theorem 3, each (G_i,w) admits a t/k-improper k-colouring c_i with colours $\{(i-1)k+1,\ldots,ik\}$. The union of the c_i's is then a t/k-improper kp-colouring of (G,w). □

Theorem 4 and its proof suggest that to find a kp-colouring with small impropriety, it may be convenient to first find a p-colouring with small impropriety and then to refine it. In addition, such a strategy, allows to adapt dynamically the refinement. In the above proof, the vertex set of each part G_i is again partitionned into k parts. However, sometimes, we shall get a better kp-colouring by partitionning each G_i into a number k_i parts, with $\sum_{i=1}^{p} k_i = kp$. Doing so, we obtain a T-improper kp-colouring of (G,w), where $T = \max\{\frac{\Delta(G_i,w)}{k_i}, 1 \leq i \leq p\}$.

One can also find upper bound on the minimum k-threshold by considering first the $k-1$ edges of largest weight around each vertex. Let (G,w) be an edge-weighted graph, and let $v_1,\ldots,v_n$ be an ordering of the vertices of G. The edges of G may be ordered in increasing order of their weight. Furthermore, to make sure that the edges incident to any particular vertex are totally ordered, we break ties according to the label of the second vertex. Formally, we say that $v_iv_j \leq_w v_iv_{j'}$ if either $w(v_iv_j) < w(v_iv_{j'}$ or $w(v_iv_j) = w(v_iv_{j'})$ and $j < j'$. With such a partial order on the edge set, the set $E_w^k(v)$ of $k-1$ greatest edges (according to this ordering) around a vertex is uniquely defined. Observe that every edge incident to v and not in $E_w^k(v)$ is smaller than an edge of $E_k(v)$ for $\leq_w$.

Let G_w^k be the graph with vertex set $V(G)$ and edge set $\bigcup_{v \in V(G)} E_w^k(v)$. Observe that every vertex of $E_w^k(v)$ has degree at least $k-1$, but a vertex may have an arbitrarily large degree. For if any edge incident to v has a greater weight than any edge not incident to v, the degree of v in G_w^k is equal to its degree in G. However we now prove that at least one vertex has degree $k-1$.

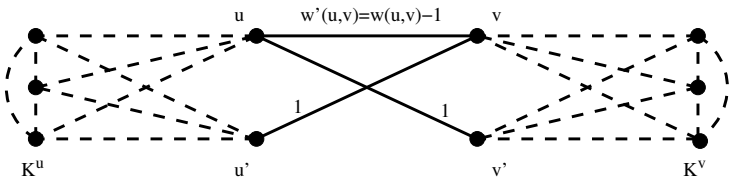


Figure 1: Construction of G^{i+1} from G^i using edge (u,v) with $k=4$. Dashed edges represent edges with infinite weights.

Proposition 5. *If (G,w) is an edge-weighted graph, then G_w^k has a vertex of degree $k-1$.*

Proof. Suppose for a contradiction, that every vertex has degree at least k, then every every vertex x there is an edge xy in $E(G_w^k)\setminus E_w^k(x)$, and so in $E_w^k(y)\setminus E_w^k(x)$. Therefore, there must be a cycle $(x_1,\dots,x_r)$ such that, for all $1\le i\le r$, $x_ix_{i+1}\in E_w^k(x_{i+1})\setminus E_w^k(x_i)$ (with $x_{r+1}=x_1$). It follows that $x_1x_2\le_w x_2x_3\le_w\cdots\le_w x_rx_1\le_w x_1x_2$. Hence, by definition, $w(x_1x_2)=w(x_2x_3)=\cdots=w(x_rx_1)=w(x_1x_2)$. Let m be the integer such that x_m has maximum index in the ordering $v_1,\dots,v_n$. Then there exists j and j' such that $x_m=v_j$ and $x_{m+2}=v_{j'}$. By definition of m, we have $j>j'$. But this contradicts the fact that $x_mx_{m+1}\le_w x_{m+1}x_{m+2}$. □

Corollary 6. *If (G,w) is an edge-weighted graph, then G_w^k has a proper k-colouring.*

Proof. By induction on the number of vertices. By Proposition 5, G_w^k has a vertex x of degree at most $k-1$. Trivially, G_w^k-x is a subgraph of $(G-x)_w^k$. By the induction hypothesis, $(G-x)_w^k$ has a proper k-colouring, which is also a proper k-colouring of $(G-x)_w^k$. This colouring can be extended in a proper k-colouring of G_w^k, by assigning to x a colour not assigned to any of its $k-1$ neighbours. □

Corollary 7. *If (G,w) is an edge-weighted graph, then $T_k(G,w)\le\Delta(G\setminus E(G_w^k),w)$.*

2.2 Transformation

In this section, we prove that the THRESHOLD IMPROPER COLOURING problem can be transformed into a problem mixing proper and improper colouring. More precisely, we prove the following:

Theorem 8. *Let (G_0,w_0) be an edge-weighted graph where w_0 is an integer-valued function, and let k be a positive integer. We can construct an edge-weighted graph (G^*,w^*) such that $w^*(e)\in\{1,M\}$ for any $e\in E(G^*)$, satisfying $T_k(G_0,w_0)=T_k(G^*,w^*)$, where $M=1+\sum_{e\in E(G)}w_0(e)$.*

Proof. Consider the function $f(G,w)=\sum_{\{e\in E(G)|w(e)\neq M\}}(w(e)-1)$.

If $f(G,w)=0$, all edges have weight either 1 or M and G has the desired property. In this case, $G^*=G$. Otherwise, we construct a graph G' and a function w' such that $T_k(G',w')=T_k(G,w)$, but $f(G',w')=f(G,w)-1$. By repeating this operation $f(G_0,w_0)$ times we get the required edge-weighted graph (G^*,w^*).

In case $f(G,w)>0$, there exists an edge $e=uv\in E(G)$ such that $2\le w(e)<M$. G' is obtained from G by adding two complete graphs on $k-1$ vertices K^u and K^v and two new vertices u' and v'. We join u and u' to all the vertices of K^u and v and v' to all the vertices of K^v. We assign weight M to all these edges. Note that, u and u' (v and v') always have the same colour, namely the remaining colour not used in K^u (resp. K^v).

We also add two edges uv' and $u'v$ both of weight 1. The edges of G keep their weight in G', except the edge $e=uv$ whose weight is decreased by one unit, i.e., $w'(e)=w(e)-1$. Thus, $f(G')=f(G)-1$ as we added only edges of weights 1 and M and we decreased the weight of e by one unit.

Now consider a weighted t-improper k-colouring c of (G,w). We produce a weighted t-improper k-colouring c' of (G',w') as follows: we keep the colours of all the vertices in G, we assign to u' (v') the same colour as u (resp. v), and we assign to K^u (resp. K^v) the $k-1$ colours different from the one used in u (resp. v).

Conversely, from any weighted improper k-colouring c' of (G',w'), we get a weighted improper k-colouring c of (G,w) by just keeping the colours of the vertices that belong to G.

For such colourings c and c' we have that $I_x(G,w,c) = I_x(G',w',c')$, for any vertex x of G different from u and v. For $x \in K^u \cup K^v$, $I_x(G',w',c') = 0$. The neighbours of u with the same colour as u in G' are the same as in G, except possibly v' which has the same colour of u if, and only if, v has the same colour of u. Let $\varepsilon = 1$ if v has the same colour as u, otherwise $\varepsilon = 0$. As the weight of uv decreases by one and we add the edge uv' of weight 1 in G', we get $I_u(G',w',c') = I_u(G,w,c) - \varepsilon + w'(u,v')\varepsilon = I_u(G,w,c)$. Similarly, $I_v(G',w',c') = I_v(G,w,c)$. Finally, $I_{u'}(G',w',c') = I_{v'}(G',w',c') = \varepsilon$. But $I_u(G',w',c') \geq (w(u,v)-1)\varepsilon$ and so $I_{u'}(G',w',c') \leq I_u(G',w',c')$ and $I_{v'}(G',w',c') \leq I_v(G',w',c')$. In summary, we have

$$\max_x I_x(G',w',c') = \max_x I_x(G,w,c)$$

and therefore $T_k(G,w) = T_k(G',w')$. □

In the worst case, the number of vertices of G^* is $n + m(w_{max}-1)2k$ and the number of edges of G^* is $m + m(w_{max}-1)[(k+4)(k-1)+2]$ with $n = |V(G)|$, $m = |E(G)|$ and $w_{max} = \max_{e \in E(G)} w(e)$.

In conclusion, this construction allows to transform the THRESHOLD IMPROPER COLOURING problem into a problem mixing proper and improper colouring. Therefore the problem consists in finding the minimum l such that a (non-weighted) l-improper k-colouring of G^* exists with the constraint that some subgraphs of G^* must admit a proper colouring. The equivalence of the two problems is proved here only for integers weights, but it is possible to adapt the transformation to prove it for rational weights.

3 Squares of Particular Graphs

As mentioned in the introduction, WEIGHTED IMPROPER COLOURING is motivated by networks of antennas similar to grids [2]. In these networks, the noise generated by an antenna undergoes an attenuation with the distance it travels. It is often modelled by a decreasing function of d, typically $1/d^\alpha$ or $1/(2^{d-1})$.

Here we consider a simplified model where the noise between two neighbouring antennas is normalised to 1, between antennas at distance two is 1/2 and 0 when the distance is strictly greater than 2. Studying this model of interference corresponds to study the WEIGHTED IMPROPER COLOURING of the square of the graph G, that is the graph G^2 obtained from G by joining every pair of vertices at distance 2, and to assign weights $w_2(e) = 1$, if $e \in E(G)$, and $w_2(e) = 1/2$, if $e \in E(G^2) \setminus E(G)$. Observe that in this case the interesting threshold values are the non-negative multiples of 1/2.

Figure 2 shows some examples of colouring for the square grid. In Figure 2(b), each vertex x has neither a neighbour nor a vertex at distance 2 coloured with its own colour, so $I_x(G^2,w_2,c) = 0$ and G^2 admits a weighted 0-improper 5-colouring. In Figure 2(c), each vertex x has no neighbour with its colour and at most one vertex of the same colour at distance 2. So $I_x(G^2,w_2,c) = 1/2$ and G^2 admits a weighted 0.5-improper 4-colouring.

For any $t \in \mathbb{R}_+$, we determine the weighted t-improper chromatic number for the square of infinite paths, square grids, hexagonal grids and triangular grids under the interference model w_2. We also present lower and upper bounds for $\chi_t(T^2,w_2)$, for any tree T and any threshold t.

3.1 Infinite paths and trees

In this section, we characterise the weighted t-improper chromatic number of the square of an infinite path, for all positive real t. Moreover, we present lower and upper bounds for $\chi_t(T^2,w_2)$, for a given tree T.

Theorem 9. *Let $P = (V,E)$ be an infinite path. Then,*

$$\chi_t(P^2, w_2) = \begin{cases} 3, & \text{if } 0 \leq t < 1; \\ 2, & \text{if } 1 \leq t < 3; \\ 1, & \text{if } 3 \leq t. \end{cases}$$

Proof. Let $V = \{v_i \mid i \in \mathbb{Z}\}$ and $E = \{(v_{i-1}, v_i) \mid i \in \mathbb{Z}\}$. Each vertex of P has two neighbours and two vertices at distance two. Consequently, the case $t \geq 3$ is trivial.

There is a 2-colouring c of (P^2, w_2) with maximum interference 1 by just colouring v_i with colour $(i \bmod 2) + 1$. So $\chi_t(P^2, w_2) \leq 2$ if $t \geq 1$. We claim that there is no weighted 0.5-improper 2-colouring of (P^2, w_2). By contradiction, suppose that c is such a colouring. If $c(v_i) = 1$, for some $i \in \mathbb{Z}$, then $c(v_{i-1}) = c(v_{i+1}) = 2$ and $c(v_{i-2}) = c(v_{i+2}) = 1$. This is a contradiction because v_i would have interference 1.

Finally, the colouring $c(v_i) = (i \bmod 3) + 1$, for every $i \in \mathbb{Z}$, is a feasible weighted 0-improper 3-colouring. □

Theorem 10. *Let $T = (V,E)$ be a tree. Then,* $\left\lceil \frac{\Delta(T) - \lfloor t \rfloor}{2t+1} \right\rceil + 1 \leq \chi_t(T^2, w_2) \leq \left\lceil \frac{\Delta(T)-1}{2t+1} \right\rceil + 2$.

Proof. The lower bound is obtained by two simple observations. First, $\chi_t(H,w) \leq \chi_t(G,w)$, for any $H \subseteq G$. Let T be a tree and v be a node of maximum degree in T. Then, observe that the weighted t-improper chromatic number of the subgraph of T^2 induced by v and its neighbourhood is at least $\lceil \frac{\Delta(T) - \lfloor t \rfloor}{2t+1} \rceil + 1$. Indeed, the colour of v can be assigned to at most $\lfloor t \rfloor$ vertices on its neighbourhood. Any other colour used in the neighbourhood of v cannot appear in more than $2t+1$ vertices because each pair of vertices in the neighbourhood of v is at distance two.

Let us look now at the upper bound. Choose any node $r \in V$ to be the root of T. Colour r with colour 1. Then, by a pre-order traversal in the tree, for each visited node v colour all the children of v with the $\lceil \frac{\Delta(T)-1}{2t+1} \rceil$ colours different from the ones assigned to v and to its parent in such a way that at most $2t+1$ nodes have the same colour. This is a feasible weighted t-improper k-colouring of T^2, with $k \leq \lceil \frac{\Delta(T)-1}{2t+1} \rceil + 2$, since each vertex interferes with at most $2t$ vertices at distance two which are children of its parent. □

3.2 Grids

In this section, we show the optimal values of $\chi_t(G^2, w_2)$, whenever G is an infinite square, hexagonal or triangular grid, for all the possible values of t.

3.2.1 Square Grid

The square grid is the graph $\mathfrak{S}$ in which the vertices are all integer linear combinations $ae_1 + be_2$ of the two vectors $e_1 = (1,0)$ and $e_2 = (0,1)$, for any $a, b \in \mathbb{Z}$. Each vertex (a,b) has four neighbours: its *down neighbour* $(a, b-1)$, its *up neighbour* $(a, b+1)$, its *right neighbour* $(a+1, b)$ and its *left neighbour* $(a-1, b)$ (see Figure 2(a)).

Theorem 11.

$$\chi_t(\mathfrak{S}^2, w_2) = \begin{cases} 5, & \text{if } t = 0; \\ 4, & \text{if } t = 0.5; \\ 3, & \text{if } 1 \leq t < 3; \\ 2, & \text{if } 3 \leq t < 8; \\ 1, & \text{if } 8 \leq t. \end{cases}$$

Proof. If $t = 0$, then the colour of vertex (a,b) must be different from the ones used on its four neighbours. Moreover, all the neighbours have different colours, as each pair of neighbours is at distance two. Consequently, at least 5 colours are needed. The following construction provides a weighted 0-improper 5-colouring of $(\mathfrak{S}^2, w_2)$:

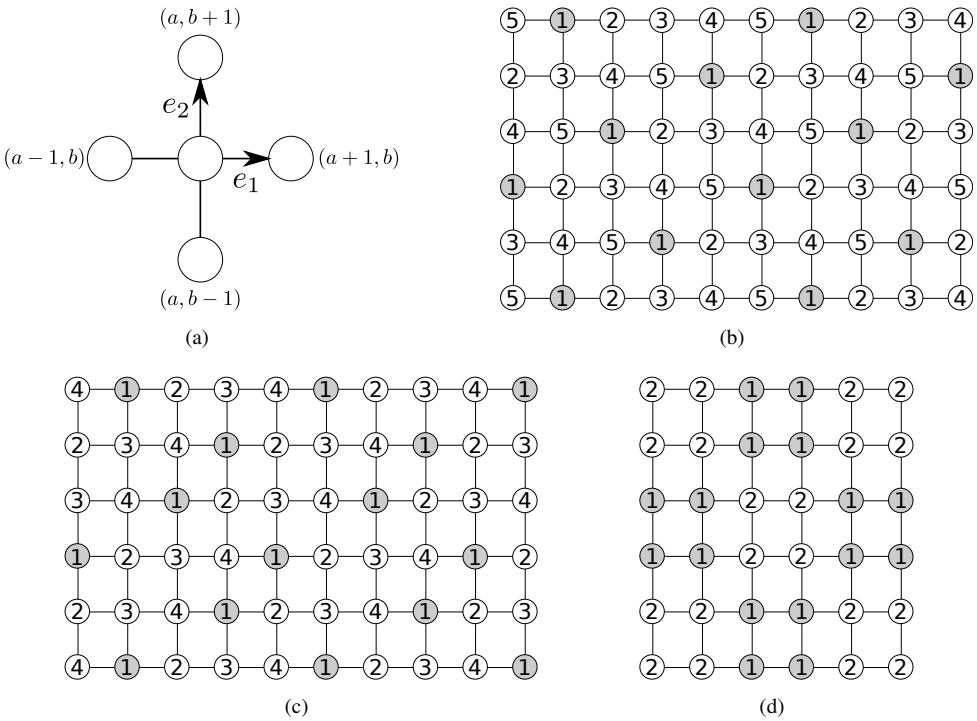


Figure 2: Optimal colourings of $(\mathfrak{S}^2, w_2)$: (b) weighted 0-improper 5-colouring of $(\mathfrak{S}^2, w_2)$, (c) weighted 0.5-improper 4-colouring of $(\mathfrak{S}^2, w_2)$, and (d) weighted 3-improper 2-colouring of $(\mathfrak{S}^2, w_2)$.

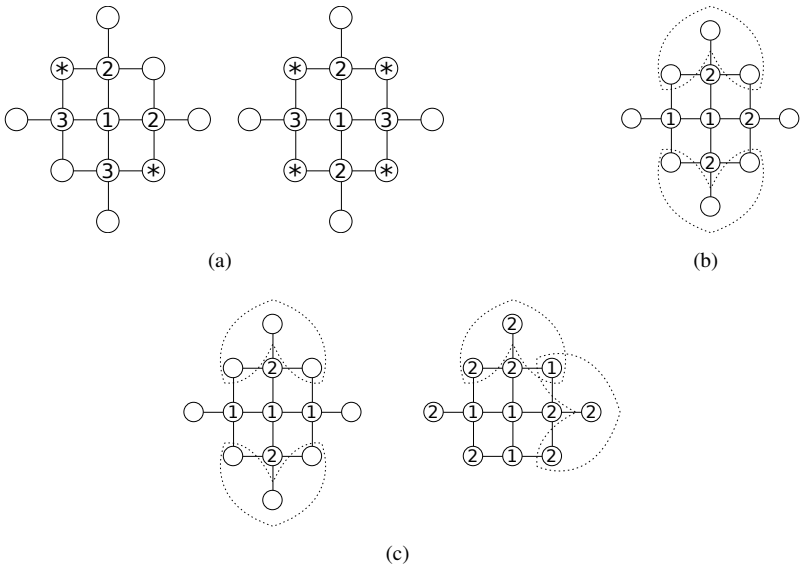


Figure 3: Lower bounds for the square grid: (a) if $t \leq 0.5$ and $k \leq 3$, there is no weighted t-improper k-colouring of $(\mathfrak{S}^2, w_2)$; (b) the first case when $t \leq 2.5$ and $k \leq 2$, and (c) the second case.

for $0 \leq j \leq 4$, let $A_j = \{(j,0) + a(5e_1) + b(2e_1 + 1e_2) \mid \forall a,b \in \mathbb{Z}\}$. For $0 \leq j \leq 4$, assign the colour $j+1$ to all the vertices in A_j (see Figure 2(b)).

When $t = 0.5$, we claim that at least four colours are needed to colour $(\mathfrak{S}^2, w_2)$. The proof is by contradiction. Suppose that there exists a weighted 0.5-improper 3-colouring of it. Let (a,b) be a vertex coloured 1. None of its neighbours is coloured 1, otherwise (a,b) has interference 1. If three neighbours have the same colour, then each of them will have interference 1. So two of its neighbours have to be coloured 2 and the two other ones 3 (see Figure 3(a)). Consider now the four nodes $(a-1,b-1)$, $(a-1,b+1)$, $(a+1,b-1)$ and $(a+1,b+1)$. For all configurations, at least two of these four vertices have to be coloured 1 (the ones indicated by a * in Figure 3(a)). But then (a,b) will have interference at least 1, a contradiction. A weighted 0.5-improper 4-colouring of $(\mathfrak{S}^2, w_2)$ can be obtained as follows (see Figure 2(c)): for $0 \leq j \leq 3$, let $B_j = \{(j,0) + a(4e_1) + b(3e_1 + 2e_2) \mid \forall a,b \in \mathbb{Z}\}$ and $B'_j = \{(j+1,2) + a(4e_1) + b(3e_1 + 2e_2) \mid \forall a,b \in \mathbb{Z}\}$. For $0 \leq j \leq 3$, assign the colour $j+1$ to all the vertices in B_j and in B'_j.

If $t = 1$, there exists a weighted 1-improper 3-colouring of $(\mathfrak{S}^2, w_2)$ given by the following construction: for $0 \leq j \leq 2$, let $C_j = \{(j,0) + a(3e_1) + b(e_1 + e_2) \mid \forall a,b \in \mathbb{Z}\}$. For $0 \leq j \leq 2$, assign the colour $j+1$ to all the vertices in C_j.

Now we prove by contradiction that for $t = 2.5$ we still need at least three colours in a weighted 2.5-improper colouring of $(\mathfrak{S}^2, w_2)$. Consider a weighted 2.5-improper 2-colouring of $(\mathfrak{S}^2, w_2)$ and let (a,b) be a vertex coloured 1. Vertex (a,b) has at most two neighbours of colour 1, otherwise it will have interference 3. We distinguish three cases:

1. Exactly one of its neighbours is coloured 1; let $(a-1,b)$ be this vertex. Then, the three other neighbours are coloured 2 (see Figure 3(b)). Consider the two sets of vertices $\{(a-1,b-1),(a+1,b-1),(a,b-2)\}$ and $\{(a-1,b+1),(a+1,b+1),(a,b+2)\}$ (these sets are surrounded by dotted lines in Figure 3(b)); each of them has at least two vertices coloured 1, otherwise the vertex $(a,b-1)$ or $(a,b+1)$ will have interference 3. But then (a,b) having four vertices at distance 2 coloured 1 has interference 3, a contradiction.

2. Two neighbours of (a,b) are coloured 1.

   (a) These two neighbours are opposite, say $(a-1,b)$ and $(a+1,b)$ (see Figure 3(c) left). Consider again the two sets $\{(a-1,b-1),(a+1,b-1),(a,b-2)\}$ and $\{(a-1,b+1),(a+1,b+1),(a,b+2)\}$ (these sets are surrounded by dotted lines in Figure 3(c) left); they both contain at least one vertex of colour 1 and therefore (a,b) will have interference 3, a contradiction.

   (b) The two neighbours of colour 1 are of the form $(a,b-1)$ and $(a-1,b)$ (see Figure 3(c) right). Consider the two sets of vertices $\{(a+1,b-1),(a+1,b+1),(a+2,b)\}$ and $\{(a+1,b+1),(a-1,b+1),(a,b+2)\}$ (these sets are surrounded by dotted lines in Figure 3(c) right); these two sets contain at most one vertex of colour 1, otherwise (a,b) will have interference 3. So vertices $(a+1,b-1)$, $(a+2,b)$, $(a,b+2)$ and $(a-1,b+1)$ are of colour 2. Vertex $(a+1,b+1)$ is of colour 1, otherwise $(a+1,b)$ has interference 3. But then $(a-2,b)$ and $(a-1,b-1)$ are of colour 2, otherwise (a,b) will have interference 3. Thus, vertex $(a-1,b)$ has exactly one neighbour coloured 1 and we are again in Case 1.

3. All neighbours of (a,b) are coloured 2. If one of these neighbours has itself a neighbour (distinct from (a,b)) of colour 2, we are in Case 1 or 2 for this neighbour. Therefore, all vertices at distance two from (a,b) have colour 1 and the interference in (a,b) is 4, a contradiction.

A weighted 3-improper 2-colouring of $(\mathfrak{S}^2, w_2)$ can be obtained as follows: a vertex of the grid (a,b) is coloured with colour $(\lfloor \frac{a}{2} \rfloor + \lfloor \frac{b}{2} \rfloor \mod 2) + 1$, see Figure 2(d).

Finally, since each vertex has four neighbours and eight vertices at distance two, there is no weighted 7.5-improper 1-colouring of $(\mathfrak{S}^2, w_2)$ and, whenever $t \geq 8$, one colour suffices.

□

3.2.2 Hexagonal Grid

There are many ways to define the system of coordinates of the hexagonal grid. Here, we use grid coordinates as shown in Figure 4. The hexagonal grid graph is then the graph $\mathfrak{H}$ whose vertex set consists in the pairs of integers $(a,b) \in \mathbb{Z}^2$ and where each vertex (a,b) has three neighbours: $(a-1,b)$, $(a+1,b)$, and $(a,b+1)$ if $a+b$ is odd, or $(a,b-1)$ otherwise.

Theorem 12.

$$\chi_t(\mathfrak{H}^2, w_2) = \begin{cases} 4, & \text{if } 0 \leq t < 1; \\ 3, & \text{if } 1 \leq t < 2; \\ 2, & \text{if } 2 \leq t < 6; \\ 1, & \text{if } 6 \leq t. \end{cases}$$

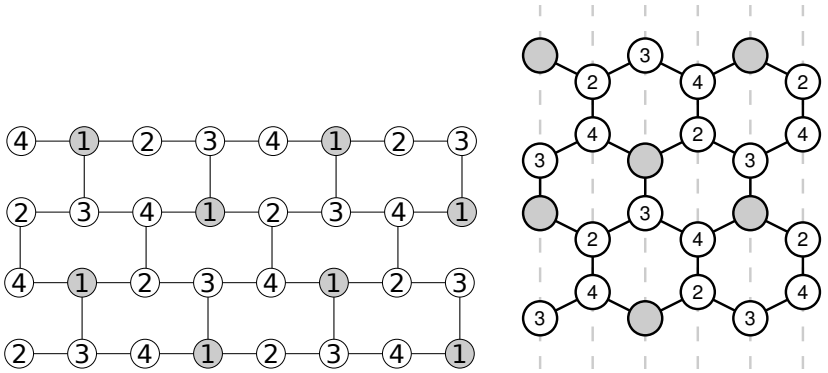
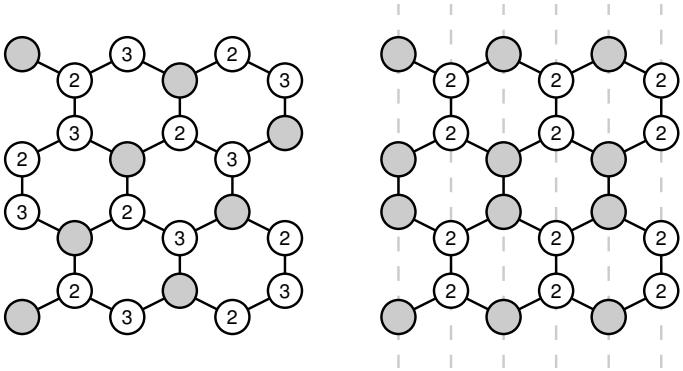


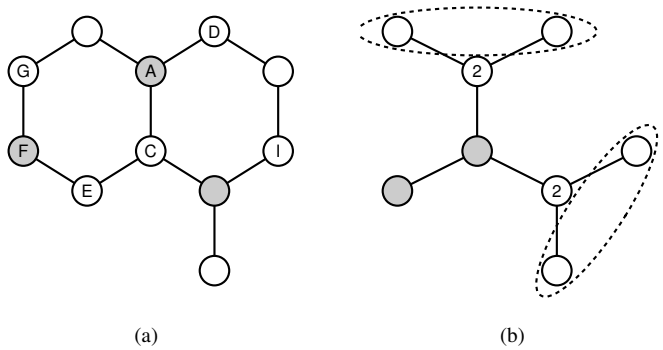
Figure 4: Weighted 0-improper 4-colouring of $(\mathfrak{H}^2, w_2)$. Left: Graph with coordinates. Right: Corresponding hexagonal grid in the euclidean space.



(a) $t = 1, k = 3$ (b) $t = 2, k = 2$

Figure 5: (a) weighted 1-improper 3-colouring of $(\mathfrak{H}^2, w_2)$ and (b) weighted 2-improper 2-colouring of $(\mathfrak{H}^2, w_2)$.

Proof. Note first, that when $t = 0$, at least four colours are needed to colour the grid, because a vertex and its neighbourhood in $\mathfrak{H}$ form a clique of size 4 in $\mathfrak{H}^2$. The same number of colours are needed if we allow a threshold $t = 0.5$. To prove this fact, let A be a vertex (a,b) of $\mathfrak{H}$ and $B = (a-1,b)$, $C = (a,b-1)$ and $D = (a+1,b)$ be its neighbours in $\mathfrak{H}$. Denote by $G = (a-2,b)$, $E = (a-1,b-1)$, $F = (a-2,b-1)$, $H = (a+1,b-1)$, $I = (a+2,b-1)$ and $J = (a+1,b-2)$ (see Figure 6(a)). By contradiction, suppose there exists a weighted 0.5-improper 3-colouring of $\mathfrak{H}^2$. Consider a node A coloured 1. Its neighbours B, C, D cannot be coloured 1 and they



(a) (b)

Figure 6: Lower bounds for the hexagonal grid. (a) when $t \leq 0.5$ and $k \leq 3$, there is no weighted t-improper k-colouring of $(\mathfrak{H}^2, w_2)$; (b) vertices coloured 2 force a vertex coloured 1 in each ellipse, leading to interference 2 in central node.

cannot all have the same colour. Without loss of generality, suppose that two of them B and C have colour 2 and D has colour 3. Then E, F and G cannot be coloured 2 because of the interference constraint in B and C. If F is coloured 3, then G and E are coloured 1, creating interference 1 in A. So F must be coloured 1 and G and E must be coloured 3. Then, H can be neither coloured 2 (interference in C) nor 3 (interference in E). So H is coloured 1. The vertex I is coloured 3, otherwise the interference constraint in H or in C is not satisfied. Then, J can receive neither colour 1, because of the interference in H, nor colour 2, because of the interference in C, nor colour 3, because of the interference in I.

There exists a construction attaining this bound and the number of colours, i.e., a 0-improper 4-colouring of $(\mathfrak{H}^2, w_2)$ as depicted in Figure 4. We define for $0 \leq j \leq 3$ the sets of vertices $A_j = \{(j,0) + a(4e_1) + b(2e_1 + e_2) | \forall a,b \in \mathbb{Z}\}$. We then assign the colour $j+1$ to the vertices in A_j. This way no vertex experiences any interference as vertices of the same colours are at distance at least three.

For $t = 1.5$ it is not possible to colour the grid with less than three colours. By contradiction, suppose that there exists a weighted 1.5-improper 2-colouring. Consider a vertex A coloured 1. If all of its neighbours are coloured 2, they have already interference 1, so all the vertices at distance two from A need to be coloured 1; this gives interference 3 in A. Therefore one of A's neighbours, say D, has to be coloured 1 and consider that the other two neighbours B and C are coloured 2. B and C have at most one neighbour of colour 2. It implies that A has at least two vertices at distance 2 coloured 1. This is a contradiction, because the interference in A would be 2 (see Figure 6(b)). Figure 5(a) presents a weighted 1-improper 3-colouring of $(\mathfrak{H}^2, w_2)$. To obtain this colouring, let $B_j = \{(j,0) + a(3e_1) + b(e_1 + e_2) \mid \forall a,b \in \mathbb{Z}\}$, for $0 \leq j \leq 2$. Then, we colour all the vertices in the set B_j with colour $j+1$, for every $0 \leq j \leq 2$.

For $t < 6$, it is not possible to colour the grid with one colour. As a matter of fact, each vertex has three neighbours and six vertices at distance two in $\mathfrak{H}$. Using one colour leads to an interference of 6. There exists a 2-improper 2-colouring of the hexagonal grid as depicted in Figure 5(b). We define for $0 \leq j \leq 1$ the sets of vertices $C_j = \{(j,0) + a(2e_1) + be_2 | \forall a,b \in \mathbb{Z}\}$. We then assign the colour $j+1$ to the vertices in C_j.

□

3.2.3 Triangular Grid

The triangular grid is the graph $\mathfrak{T}$ whose vertices are all the integer linear combinations $af_1 + bf_2$ of the two vectors $f_1 = (1,0)$ and $f_2 = (\frac{1}{2}, \frac{\sqrt{3}}{2})$. Thus we may identify the vertices with the ordered pairs (a,b) of integers. Each vertex $v = (a,b)$ has six neighbours: its *right neighbour* $(a+1,b)$, its *right-up neighbour* $(a,b+1)$, its

left-up neighbour $(a-1,b+1)$, its *left neighbour* $(a-1,b)$, its *left-down neighbour* $(a,b-1)$ and its *right-down neighbour* $(a+1,b-1)$ (see Figure 7(a)).

Theorem 13.

$$\chi_t(\mathfrak{T}^2,w_2)=\begin{cases}7, & \text{if } t=0;\\ 6, & \text{if } t=0.5;\\ 5, & \text{if } t=1;\\ 4, & \text{if } 1.5\leq t<3;\\ 3, & \text{if } 3\leq t<5;\\ 2, & \text{if } 5\leq t<12;\\ 1, & \text{if } 12\leq t.\end{cases}$$

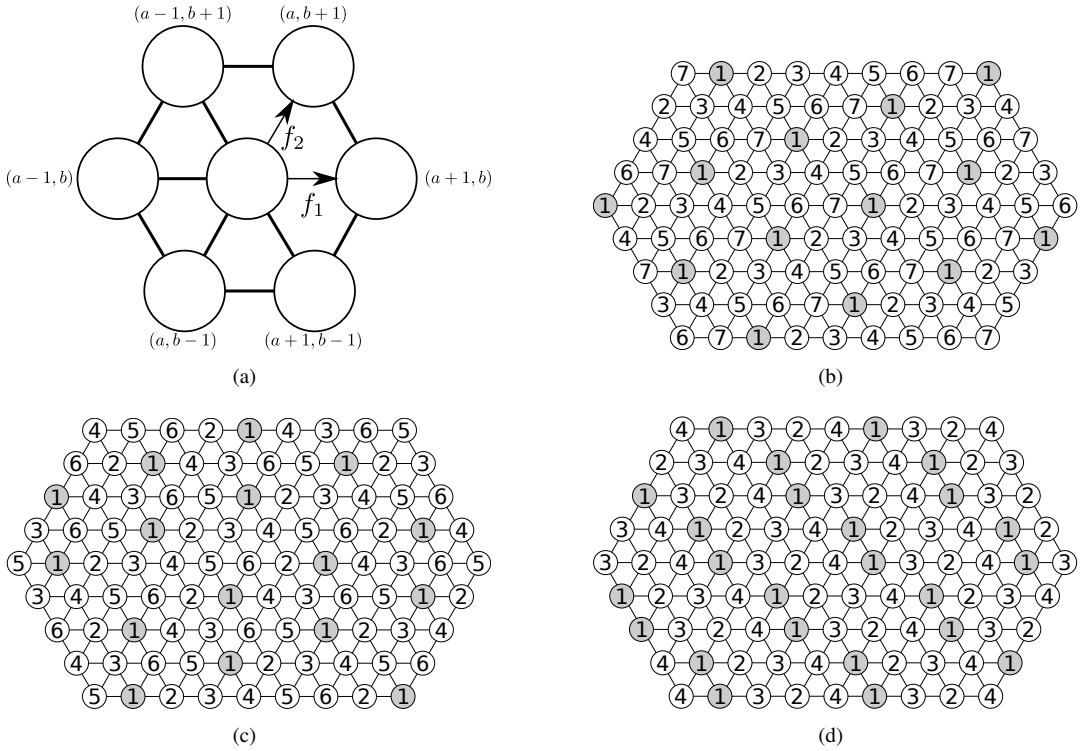


Figure 7: Optimal colourings of $(\mathfrak{T}^2,w_2)$: (b) weighted 0-improper 7-colouring of $(\mathfrak{T}^2,w_2)$, (c) weighted 0.5-improper 6-colouring of $(\mathfrak{T}^2,w_2)$, and (d) weighted 1.5-improper 4-colouring of $(\mathfrak{T}^2,w_2)$

Proof. If $t=0$, there is no weighted 0-improper 6-colouring of $(\mathfrak{T}^2,w_2)$, since in $\mathfrak{T}^2$ there is a clique of size 7 induced by each vertex and its neighbourhood. There is a weighted 0-improper 7-colouring of $(\mathfrak{T}^2,w_2)$ as depicted in Figure 7(b). This colouring can be obtained by the following construction: for $0\leq j\leq 6$, let $A_j=\{(j,0)+a(7f_1)+b(2f_1+f_2)\mid \forall a,b\in\mathbb{Z}\}$. For $0\leq j\leq 6$, assign the colour $j+1$ to all the vertices in A_j.

In what follows, we denote by V_0 a vertex coloured 1; by N_0, N_1, N_2, N_3, N_4, N_5 the six neighbours of V_0 in $\mathfrak{T}$ be in a cyclic order. Let Γ^2 be the set of twelve vertices at distance 2 of V_0 in $\mathfrak{T}$; more precisely $N_{i(i+1)}$ denotes the vertex of Γ^2 adjacent to both N_i and N_{i+1} and by N_{ii} the vertex of Γ^2 joined only to N_i, for every $i\in\{0,\ldots,5\}$, $i+1$ is taken modulo 6 (see Figure 8).

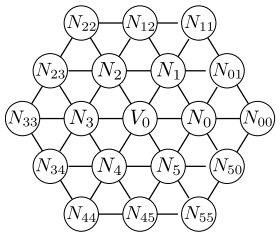


Figure 8: Notation used in proofs of non-existence of weighted improper colourings of $(\mathfrak{T}^2, w_2)$.

We claim that there is no weighted 0.5-improper 5-colouring of $(\mathfrak{T}^2, w_2)$. We prove it by contradiction, thus let c be such a colouring. No neighbour of V_0 can be coloured 1, otherwise $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 1$. As two consecutive neighbours are adjacent, they cannot have the same colour. Furthermore, there cannot be three neighbours with the same colour (each of them will have an interference at least 1). As there are four colours different from 1, at least two of them, say 2 and 3, are repeated twice among the six neighbours. So, there exists a sequence of three consecutive neighbours the first one with a colour different from 2 and 3 and the two others coloured 2 and 3. W.l.o.g., let $c(N_5) = 4$, $c(N_0) = 2$, $c(N_1) = 3$.

Note that the vertices coloured 2 and 3 have already an interference of 0.5, and so none of their vertices at distance 2 can be coloured 2 or 3. In particular, let $A = \{N_{50}, N_{00}, N_{01}, N_{11}, N_{12}\}$; the vertices of A cannot be coloured 2 or 3. At most one vertex in Γ^2 can be coloured 1, otherwise $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 1$. If there is no vertex coloured 1 in A, we have a contradiction as we cannot have a sequence of five vertices uniquely coloured 4 and 5 (indeed colours should alternate and the vertex in the middle N_{01} will have interference at least 1). Suppose N_4 is coloured 3, then N_{45} and N_{55} can only be coloured 1 and 5; but, as they have different colours, one is coloured 1 and so there is no vertex coloured 1 in A. So the second vertex coloured 3 in the neighbourhood of V_0 is necessarily N_3 (it cannot be N_2 neighbour of N_1 coloured 3). Then, N_4 cannot be also coloured 5, otherwise N_{45} is coloured 1 and again there is no vertex coloured 1 in A. In summary $c(N_4) = 2$, $c(N_3) = 3$ and the vertex of Γ^2 coloured 1 is in A. But then the five consecutive vertices $A' = \{N_{23}, N_{33}, N_{34}, N_{44}, N_{45}\}$ can only be coloured 4 and 5. A contradiction as $I_{N_{34}}(\mathfrak{T}^2, w_2, c) \geq 1$.

A weighted 0.5-improper 6-colouring of $(\mathfrak{T}^2, w_2)$ can be obtained by the following construction (see Figure 7(c)): for $0 \leq j \leq 11$, let $B_j = \{(j,0) + a(12f_1) + b(2f_1 + f_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 5$, assign the colour $j+1$ to all the vertices in B_j, B_6 with colour 2, B_7 with colour 1, B_8 with colour 4, B_9 with colour 3, B_{10} with colour 6 and B_{11} with colour 5.

Now we prove that $(\mathfrak{T}^2, w_2)$ does not admit a weighted 1-improper 4-colouring. Again, by contradiction, suppose that there exists a weighted 1-improper 4-colouring c of $(\mathfrak{T}^2, w_2)$. We analyse some cases:

1) There exist two adjacent vertices in $\mathfrak{T}$ with the same colour.

   Let V_0 and one of its neighbours be both coloured 1. Note that no other neighbour of V_0, nor the vertices at distance 2 from V_0 are coloured 1 (otherwise, $I_{V_0}(\mathfrak{T}^2, w_2, c) > 1$). We use intensively the following facts:

   Fact 1 (F_3)**.** *There do not exist three consecutive vertices with the same colour (otherwise the vertex in the middle would have interference at least 2).*

   Fact 2 (F_5)**.** *In a sequence of five consecutive vertices there cannot be four of the same colour.*

   One colour other than 1 should appear at least twice. Let this colour be denoted 2 (the other colours being denoted 3 and 4).

   a) Two neighbours of V_0 coloured 2 are consecutive, say N_0 and N_1. By Fact F_3, N_2 is coloured 3. None of N_{01}, N_{11}, N_{12}, N_{22} and N_{23} can be coloured 2, otherwise $I_{N_1}(\mathfrak{T}^2, w_2, c) > 1$. One of N_{12}, N_{22} and N_{23} is coloured

3, otherwise we contradict Fact F_3 with colour 4 and at most one of N_{01}, N_{11}, N_{12}, N_{22} and N_{23} is coloured 3, otherwise $I_{N_2}(\mathfrak{T}^2, w_2, c) > 1$; but we have a contradiction with Fact F_5.

b) Two neighbours of V_0 coloured 2 are at distance 2, say N_0 and N_2. Then N_{50}, N_{00} and N_{01} (respectively N_{12}, N_{22} and N_{23}) are not coloured 2, otherwise $I_{N_0}(\mathfrak{T}^2, w_2, c) > 1$ (respectively $I_{N_2}(\mathfrak{T}^2, w_2, c) > 1$). One of N_3 and N_5 is not coloured 1, say N_3. It is not coloured 2, $I_{N_3}(\mathfrak{T}^2, w_2, c) > 1$. Let $c(N_3) = 3$. If N_4 or N_{11} is coloured 2, then N_{33} and N_{34} are not coloured 2, otherwise $I_{N_2}(\mathfrak{T}^2, w_2, c) > 1$ and we have a sequence of 5 vertices N_{12}, N_{22}, N_{23}, N_{33} and N_{34} contradicting Fact F_5 as four are of colour 4 (indeed, at most one is coloured 3 due to interference in colour 3 with N_3 or N_{22}). So N_{11} is coloured 3 or 4. If N_1 also is coloured 3 or 4, we have a contradiction with Fact F_5 applied to the five vertices N_{00}, N_{01}, N_{11}, N_{12} and N_{22}, by the same previous argument. So $c(N_1) = 1$; furthermore N_4 is not coloured 1 (at most one neighbour coloured 1), nor 2 as we have seen above, nor 3, otherwise we are in the case (a). Therefore $c(N_4) = 4$ and $c(N_5) = 3$, by the same reason. But then $c(N_{23}) = 4$, otherwise the interference in V_0 or N_2 or N_3 is greater than 1. N_{33} and N_{34} can be only coloured 2, otherwise V_0, N_3, N_4 or N_{23} will have interference stricly greater than 1, but N_{33} has interference greater than 1, a contradiction.

c) Two neighbours of V_0 coloured 2 are at distance 3 say N_0 and N_3. Then N_{50}, N_{00} and N_{01} (respectively N_{23}, N_{33} and N_{34}) are not coloured 2, otherwise $I_{N_0}(\mathfrak{T}^2, w_2, c) > 1$ (respectively $I_{N_3}(\mathfrak{T}^2, w_2, c) > 1$). W.l.o.g., let N_1 be the vertex coloured 1. Among the four vertices N_{12}, N_{22}, N_{44} and N_{45} at most one is coloured 2, otherwise $I_{N_3}(\mathfrak{T}^2, w_2, c) > 1$). So, without loss of generality, we can suppose N_{44} and N_{45} are coloured 3 or 4; but we have a set of five consecutive vertices N_{23}, N_{33}, N_{34}, N_{44}, N_{45}, contradicting Fact F_5 (indeed at most one can be of the colour of N_4).

2) No colour appears in two adjacent vertices of $\mathfrak{T}$.

Let V_0 be coloured 1. No colour can appear four or more times among the neighbours, otherwise there are two neighbours with the same colour.

a) One colour appears three times among the neighbours of V_0, say $c(N_0) = c(N_2) = c(N_4) = 2$. W.l.o.g., let $c(N_1) = 3$. No vertex at distance 2 can be coloured 2. N_{01}, N_{11} and N_{12} being neighbours of N_1 cannot be coloured 3 and they cannot be all coloured 4. So one of N_{01}, N_{11}, N_{12} is coloured 1. Similarly one of N_{23}, N_{33}, N_{34} is coloured 1 (same reasoning with N_3 instead of N_1) and one of N_{45}, N_{55}, N_{50} is coloured 1, so $I_{V_0}(\mathfrak{T}^2, w_2, c) > 1$.

b) The three colours appear each exactly twice in the neighbourhood of V_0.

i) The same colour appears in some N_i and N_{i+2}, $i \in \{0,1,2,3\}$. W.l.o.g., let $c(N_0) = c(N_2) = 2$ and $c(N_1) = 3$. Then, $c(N_3) = c(N_5) = 4$ and $c(N_4) = 3$. Then, $c(N_{50}) = 1$ or 3, $c(N_{01}) = 1$ or 4. If $c(N_{50}) = 3$ and $c(N_{01}) = 4$, then $c(N_{00}) = 1$. Among N_{50}, N_{00}, N_{01}, at least one has colour 1. Similarly one of N_{12}, N_{22}, N_{23} has colour 1. So $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 1$ and $c(N_{34}) = c(N_{45}) = 2$. Consequently, no matter the colour of N_{44} some vertex will have interference greater than 1.

ii) We have $c(N_0) = c(N_3) = 2$, $c(N_1) = c(N_4) = 3$ and $c(N_2) = c(N_5) = 4$. Here one of N_{50}, N_{00}, N_{01} has colour 1 and similarly one of N_{12}, N_{22}, N_{23} has colour 1 and one of N_{34}, N_{44}, N_{45}. Therefore $I_{V_0}(\mathfrak{T}^2, w_2, c) > 1$, a contradiction.

To obtain a weighted 1-improper 5-colouring of $(\mathfrak{T}^2, w_2)$, for $0 \leq j \leq 4$, let $C_j = \{(j,0) + a(5f_1) + b(2f_1 + f_2) \mid \forall a, b \in \mathbb{Z}\}$. For $0 \leq j \leq 4$, assign the colour $j+1$ to all the vertices in C_j.

$(\mathfrak{T}^2, w_2)$ has a weighted 1.5-improper 4-colouring as depicted in Figure 7(d). Formally, this colouring can be obtained by the following construction: for $0 \leq j \leq 3$, let $D_j = \{(j,0) + a(4f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$; then assign colour D_0 with colour 4, D_1 with colour 1, D_2 with colour 3 and D_3 with colour 2. Now, for $0 \leq j \leq 3$, let $D'_j = \{(j,1) + a(4f_1) + b(f_1 + 2f_2) \mid \forall a, b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 3$, colour $j+1$ to all the vertices in D'_j.

For determining the lower bounds for the cases in which $\chi_t(\mathfrak{T}^2, w_2)$ is equal to 3, the proof involved too many subcases to be readable. Then, we used an ILP solver with the integer linear programming formulation we present

in Section 4 to validate it. We verify that the square of triangular grids with 64 vertices did not accept any weighted 2.5-improper 3-colouring.

Now we present the colouring providing the corresponding upper bound.

For a weighted 3-improper 3-colouring of $(\mathfrak{T}^2, w_2)$ set, for $0 \leq j \leq 2$, $E_j = \{(j,0) + a(3f_1) + b(f_2) \mid \forall a,b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 2$, assign the colour $j+1$ to all the vertices in E_j.

Now we prove that $(\mathfrak{T}^2, w_2)$ does not admit a weighted 4.5-improper 2-colouring. Again, by contradiction, suppose that there exists a weighted 4.5-improper 2-colouring c of $(\mathfrak{T}^2, w_2)$ with the interference function w_2. A vertex can have at most four neighbours of the same colour as it. We analyse some cases:

1) There exists a vertex V_0 with four of its neighbours coloured with its own colour, say 1. Therefore among the vertices of Γ^2 at most one is coloured 1. Consider the two neighbours of V_0 coloured 2. First, consider the case in which they are adjacent and let them be N_0 and N_1. In Γ^2, N_0 has three neighbours and four vertices at distance 2; since at most one being of colour 1, these vertices produce in N_0 an interference of 4 and as N_1 is also of colour 2, then $I_{N_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction. In case the two neighbours of V_0 coloured 2 are non adjacent, let them be N_i and N_j. At least one of them, say N_i has its three neighbours in Γ^2 coloured 2 and it has also at least three vertices at distance 2 in Γ^2 coloured 2; taking into account that N_j is coloured 2 and at distance two from N_i, we get $I_{N_i}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

2) No vertex has four neighbours with its colour and there exists at least one vertex V_0 coloured 1 that has three neighbours of the same colour 1.

   a) The three other neighbours of V_0 coloured 2 are consecutive and let them be N_0, N_1 and N_2. N_{34}, N_{44} and N_{45} are all coloured 2, otherwise N_4 will have four neighbours coloured 1 and we will be in Case 1. At most one of N_{01}, N_{11} and N_{12} has colour 2, otherwise N_1 will have four neighbours coloured 2 and we will be again in Case 1.

      i) N_{11} is coloured 2. Then $c(N_{01}) = c(N_{12}) = 1$. As already $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 4$, there is at most another vertex in Γ^2 coloured 1. So either the three vertices N_{22}, N_{23} and N_{33} or the three vertices N_{00}, N_{50} and N_{55} are all coloured 2 and then $I_{N_2}(\mathfrak{T}^2, w_2, c) \geq 5$ or $I_{N_5}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

      ii) N_{01} is coloured 2 (the case N_{12} is symmetric). Then, $c(N_{11}) = c(N_{12}) = 1$. One of N_{00} and N_{50} is of colour 1 otherwise, N_0 has four neighbours of colour 2. But then $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 4.5$ so all the other vertices of Γ^2 are coloured 2. Therefore, $I_{N_2}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

      iii) N_{01}, N_{11} and N_{12} all have colour 1. In that case $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 4.5$. Therefore all the other vertices of Γ^2 are coloured 2 and $I_{N_0}(\mathfrak{T}^2, w_2, c) \geq 4.5$. So the other vertices at distance 2 of N_0 are coloured 1 and then $I_{N_{01}}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

   b) Among the three vertices of colour 2, only two are consecutive. Without loss of generality, let the three vertices of colour 2 be N_0 and N_1 and N_3. At least one vertex of N_{50}, N_{00}, N_{01} is coloured 1, otherwise N_0 has four neighbours of the same colour as it. Similarly at least one vertex of N_{01}, N_{11}, N_{12} is coloured 1, otherwise N_1 has four neighbours with its colour. At least one vertex of N_{23}, N_{33}, N_{34} is coloured 1, otherwise N_3 has three consecutive neighbours of the same colour as it and we are in the previous case. Suppose N_{01} is coloured 2, then $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 4.5$ and exactly one of N_{50}, N_{00} and one of N_{11}, N_{12} is coloured 1 and N_{45}, N_{55} are coloured 2, otherwise $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$. Then $I_{N_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction. So, $c(N_{01}) = 1$. If both N_{50}, N_{00} are coloured 2, then $I_{N_0}(\mathfrak{T}^2, w_2, c) \geq 5$ with three neighbours coloured 2 and at least four vertices at distance 2 coloured 2, namely N_3 and three vertices among N_{45}, N_{55}, N_{11}, N_{12} (at most one vertex of these could be of colour 1, otherwise $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$). So, one of N_{50}, N_{00} is coloured 1 and all the other vertices in $\{N_{11}, N_{12}, N_{22}, N_{44}, N_{45}, N_{55}\}$ are coloured 2 implying that $I_{N_3}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

   c) No two vertices of colour 2 are consecutive. Let these vertices be N_0, N_2, N_4. The three neighbours of N_0 (resp. N_1, N_2) in Γ^2 cannot be all coloured 2, otherwise we are in Case (a). So exactly one neighbour of N_0, N_1, N_2 in Γ^2 is coloured 1, otherwise $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$. Furthermore all the other vertices of Γ^2

are all coloured 2. Then, if $c(N_{12}) = c(N_{45}) = 2$, we conclude that $I_{N_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction. Consequently, w.l.o.g., suppose that $c(N_{12}) = 1$. In this case, N_{23} has at least three neighbours coloured 2 and we are in some previous case.

3) No vertex has three neighbours coloured with its own colour, but there exists at least one vertex, say V_0, of colour 1 that has two neighbours coloured 1.

   a) These two neighbours are consecutive say N_0 and N_1. The neighbours of N_3 and N_4 in Γ^2 are all coloured 1, otherwise they would have at least three neighbours with the same colour. Similarly, at least one of N_{12} and N_{22} is coloured 1, otherwise N_2 would have at least three neighbours also coloured 2. Then, $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

   b) These two neighbours are of the form N_i and N_{i+2}, for some $i \in \{0,1,2,3\}$. W.l.o.g., let these neighbours be N_0 and N_2. Thus, the three neighbours of N_4 in Γ^2, N_{34}, N_{44} and N_{45} are coloured 1 and at least one vertex of N_{23} and N_{33} (resp. N_{55} and N_{50}) is coloured 1. Moreover, at least one vertex of N_{01}, N_{11} and N_{12} must be coloured 1, otherwise N_1 would have three neighbours with its colour. Consequently, $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

   c) These two neighbours are of the form N_i and N_{i+3}, for some $i \in \{0,1,2\}$. W.l.o.g., let these neighbours be N_0 and N_3. Again, at least three vertices among N_{01}, N_{11}, N_{12}, N_{22} and N_{23} and at least three other vertices among N_{34}, N_{44}, N_{45}, N_{55} and N_{50} are coloured 1. Consequently, $I_{V_0}(\mathfrak{T}^2, w_2, c) \geq 5$, a contradiction.

4) No vertex has two neighbours of the same colour. Suppose V_0 is coloured 1 and has only one neighbour N_0 coloured 1. Then, its other five neighbours are coloured 2 and N_2 has two neighbours of the colour 2, a contradiction.

A weighted 5-improper 2-colouring of $(\mathfrak{T}^2, w_2)$ is obtained as follows: for $0 \leq j \leq 1$, let $F_j = \{(j,0) + a(2f_1) + b(f_1 + 2f_2) \mid \forall a,b \in \mathbb{Z}\}$ and $F'_j = \{(j-1,1) + a(2f_1) + b(f_1 + 2f_2) \mid \forall a,b \in \mathbb{Z}\}$. Then, for $0 \leq j \leq 1$, assign the colour $j+1$ to all the vertices in F_j and in F'_j.

Since each vertex has six neighbours and twelve vertices at distance 2 in $\mathfrak{T}$, there is no weighted t-improper 1-colouring of $(\mathfrak{T}^2, w_2)$, for any $t < 12$. Obviously, there is a weighted 12-improper 1-colouring of $\mathfrak{T}^2$. □

4 Integer Linear Programming Formulations, Algorithms and Results

In this section, we look at how to solve the WEIGHTED IMPROPER COLOURING and THRESHOLD IMPROPER COLOURING for realistic instances. We consider Poisson-Voronoi tesselations as they are good models of antenna networks [3, 9, 10]. We present integer linear programming models for both problems. Then, we introduce two algorithmic approaches for THRESHOLD IMPROPER COLOURING: a simple greedy heuristic and a Branch-and-Bound algorithm.

4.1 Integer Linear Programming Formulations and Algorithms

4.1.1 Integer Linear Programming Models

Given an edge-weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}^*_+$, and a positive real threshold t, we model WEIGHTED IMPROPER COLOURING by using two kinds of binary variables. Variable x_{ip} indicates if vertex i is coloured p and variable c^p indicates if colour p is used, for every $1 \leq i \leq n$ and $1 \leq p \leq l$, where l is an upper bound for the number of colours needed in an optimal weighted t-improper colouring of G (see Section 2). The model follows:

$$\begin{array}{rll}
\min & \sum_{p\in\{1,\ldots,l\}} c^p & \\
\text{subject to} & & \\
& \sum_{ij\in E; j\neq i} w(i,j)x_{jp} \leq t + M(1-x_{ip}) & (\forall i \in V, \forall p \in \{1,\ldots,l\}) \\
& c^p \geq x_{ip} & (\forall i \in V, \forall p \in \{1,\ldots,l\}) \\
& \sum_{p\in\{1,\ldots,l\}} x_{ip} = 1 & (\forall i \in V) \\
& x_{ip} \in \{0,1\} & (\forall i \in V, \forall p \in \{1,\ldots,l\}) \\
& c^p \in \{0,1\} & (\forall p \in \{1,\ldots,l\})
\end{array}$$

where M is a large integer. For instance, it is sufficient to choose $M > \sum_{(u,v)\in E} w(u,v)$.

For THRESHOLD IMPROPER COLOURING, given an edge-weighted graph $G = (V,E,w)$, $w : E \to \mathbb{R}_+^*$, and a positive integer k, the model we consider is:

$$\begin{array}{rll}
\min & t & \\
\text{subject to} & & \\
& \sum_{ij\in E; j\neq i} w(i,j)x_{jp} \leq t + M(1-x_{ip}) & (\forall i \in V, \forall p \in \{1,\ldots,k\}) \\
& \sum_{p\in\{1,\ldots,k\}} x_{ip} = 1 & (\forall i \in V) \\
& x_{ip} \in \{0,1\} & (\forall i \in V, \forall p \in \{1,\ldots,k\})
\end{array}$$

4.1.2 Levelling Heuristic

We develop a heuristic to solve THRESHOLD IMPROPER COLOURING. The idea is to try to level the distribution of interference over the vertices. Each vertex is coloured one after the other by the colour minimising the local interference. This is achieved by considering for the nodes not yet coloured the "potential interference", i.e., the interference induced by the already coloured vertices.

Precisely, consider a vertex v not yet coloured and a colour $i \in \{1,\ldots,k\}$. We define the potential interference $I'_{v,i}$ as:

$$I'_{v,i} = \sum_{u\in N(v)\cap V_i} w(u,v),$$

where V_i is the set of vertices that have already been assigned the colour i. The order in which vertices are coloured is decided according to the total potential interference, defined as $I''_v = \sum_{i=1}^{k} I'_{v,i}$. The algorithm finds a feasible colouring in the first step and tries to improve it for p runs, where p is part of the input.

- The interference target is set $t_t = M$;
- while the number of runs is smaller than p;
  - all potential interferences are set to zero;
  - while there are still vertices to colour:
    - choose a vertex v randomly among the uncoloured vertices that have the maximum total potential interference;
    - try each colour i in the order of increasing potential interference $I'_{v,i}$:
      - if colouring v with i does not result in interference greater than t_t for v or any of its neighbours, colour v with i, else try a new colour;
      - if all colours resulted in excessive interferences, start new run.
  - If all the vertices were successfully coloured, set $t_t = \max_{v\in V, i\in\{1,\ldots,k\}} I_v(G,w,c) - \gcd(w)$ and store the colouring as the best found.

As a *randomised greedy colouring* heuristic, it has to be run multiple times to achieve satisfactory results. This is not a practical issue due to low computational cost of each run. The local immutable colouring decision is taken in time $O(k\Delta)$. Then, after each such decision, the interference has to be propagated, which takes time linear in the vertex degree. This gives a computational complexity bound $O(kn\Delta)$-time.

4.1.3 Branch-and-Bound Algorithm

We also implemented a simple Branch-and-Bound algorithm inspired by the above heuristic. The first step of the algorithm is to defined the order in which vertices are coloured. This order is produced by a similar procedure to the one used in the above heuristic. In order to compute this order, we start by marking a random vertex and setting it as the first in a *to colour* list. Then, as long as there are unmarked vertices, we keep choosing a random vertex u among the unmarked vertices with biggest $\sum_{v\in N(u)\cap V_m} w(u,v)$, where V_m is the set of already marked vertices. Then we mark u and append it to the *to order*, till we have processed all vertices.

A basic Branch-and-Bound colours vertices in the obtained order. Potential interference, as defined for the heuristic, is tracked with the additional step of decreasing the values when backing off from a colouring. Colours are tried in the order of increasing potential interference. Thanks to that it produces results similar to the heuristic in a short time. On the other hand it is guaranteed to find the optimal solution in a finite time.

In the following, we compare the performance of the ILP models with the one of the Levelling heuristic and the Branch-and-Bound algorithm .

4.2 Results

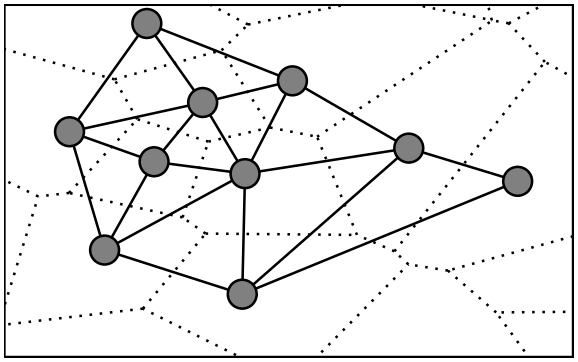
In this section, we look at the performances of the methods to solve the THRESHOLD IMPROPER COLOURING. We consider Delaunay graphs (dual of Voronoi diagram) for a set of random points, see Figure 9(a). This kind of graphs is a natural approximation of a network of irregular cells. The interference model is the one described in Section 3: adjacent nodes interfere by 1 and nodes at distance two interfere by 1/2.

Figure 9 shows a performance comparison of the above-mentioned algorithms. For all the plots, each data point represents an average over a number (between 10 and 100) of different graphs. The same graph is used for all values of colours and time limit. Therefore Figures 9(b) and 9(c) plot how results for a given problem instance get enhanced with increasing time limits. Plots 9(e) and 9(f) show decreasing interference along increasing the number of colours allowed. Finally plot 9(d) shows how well all the programmes scale with increasing graph sizes.

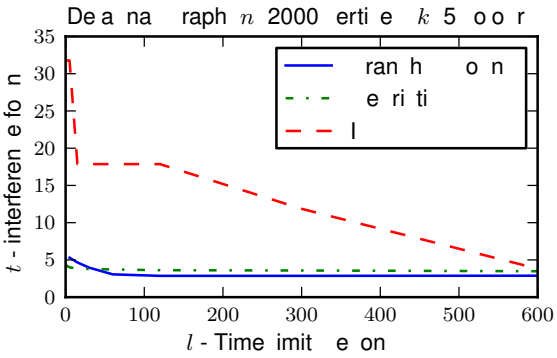
One immediate observation about both the heuristic and Branch-and-Bound algorithm is that they provide solutions in relatively short time. Despite their naive implementation in a high-level programming language, they tend to find near-optimal results in matter of seconds even for graphs of thousands of vertices. On the other hand, with limited time, they fail to improve up to optimal results, especially with a low number of allowed colours. Although it is easy to envision an implementation faster by orders of magnitude, this may still give little improvement — once a near-optimal solution is found, the Branch-and-Bound algorithm does not improve for a very long time (an example near-optimal solution found in around three minutes was not improved in over six days).

ILP solvers with good Branch-and-Cut implementations do not suffer from this problem. However, they can not take advantage of any specialised knowledge of the problem, only the basic integer linear programmming representation. Thus it takes much more time to produce first good results. Despite taking advantage of multi-core processing, CPLEX — ILP solver used in this work, does not scale with increasing graph sizes as well as our simple algorithms. Furthermore, Figure 9(e) reveals one problem specific to ILP solvers. When increasing the number of allowed colours, obtaining small interferences gets easier. But this introduces additional constraints in the integer linear programming formulation, thus increasing the complexity for a solver.

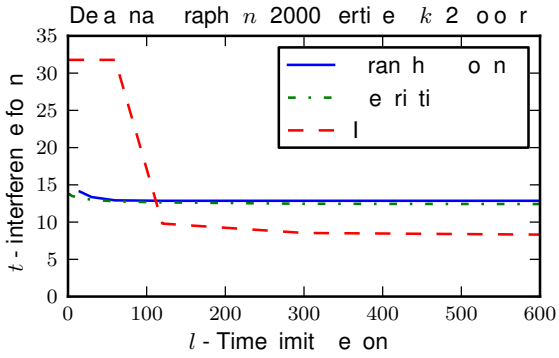
Above observations are valid only for the very particular case of the simple interference function and very sparse graphs. The average degree in Delaunay graph G converges to 6 (this results follows from the observation



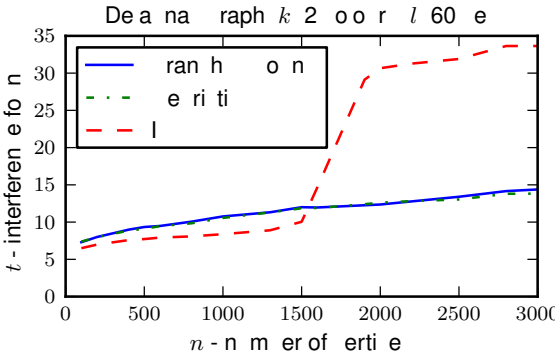
(a) Example Delaunay graph, dotted lines delimit corresponding Voronoi diagram cells



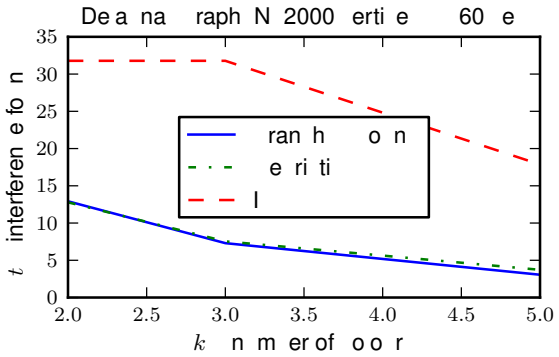
(b) Over time



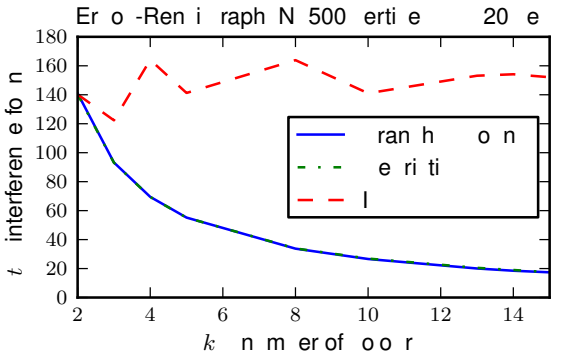
(c) Over time



(d) Over size



(e) Over colours



(f) Over colours

Figure 9: Results comparison for Levelling heuristic, Branch-and-Bound algorithm and Integer Linear Programming Formulation.

that G is planar and triangulated, thus $|E(G)| = 3|V(G)| - 6$ by Euler's formula). Proposed algorithms also work quite well for denser graphs. Figure 9(f) plots interferences for different numbers of colours allowed found by the programs for an Erdös-Rényi graph with n=500 and p=0.1. This gives us an average degree of 50. Both Branch-and-Bound and heuristic programs achieve acceptable, and nearly identical, results. But the large number of constraints makes the integer linear programming formulation very inefficient.

5 Conclusion, Open Problems and Future Directions

In this paper, we introduced and studied a new colouring problem, WEIGHTED IMPROPER COLOURING. This problem is motivated by the design of telecommunication antenna networks in which the interference between two vertices depends on different factors and can take various values. For each vertex, the sum of the interferences it receives should be less than a given threshold value.

We first give general bounds on the weighted-improper chromatic number. We then study the particular case of infinite paths, trees and grids: square, hexagonal and triangular. For these graphs, we provide their weighted-improper chromatic number for all possible values of t. Finally, we propose a heuristic and a Branch-and-Bound algorithm to find good solutions of the problem. We compare their results with the one of an integer linear programming formulation on cell-like networks, Poisson-Voronoi tessellations.

Many problems remain to be solved:

- For the study of the grid graphs, we considered a specific function where vertex at distance one interfere by 1 and vertices at distance 2 by 1/2. Other weight functions should be considered. e.g. $1/d^2$ or $1/(2^{d-1})$, where d is the distance between vertices.
- Other families of graphs could be considered, for example hypercubes.
- Let $G=(V,E,w)$ be an edge-weighted graph where the weights are all equal to 1 or M. Let G_M be the subgraph of G induced by the edges of weight M; is it true that if $\Delta(G_M) << \Delta(G)$, then $\chi_t(G,w) \leq \chi_t(G) \leq \left\lceil \frac{\Delta(G,w)+1}{t+1} \right\rceil$? A similar result for $L(p,1)$-labelling [11] suggests it could be true.

References

[1] K.I. Aardal, S.P.M. van Hoesel, A.M.C.A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007.

[2] S. Alouf, E. Altman, J. Galtier, J.F. Lalande, and C. Touati. Quasi-optimal bandwidth allocation for multi-spot MFTDMA satellites. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 560–571. IEEE, 2005.

[3] F. Baccelli, M. Klein, M. Lebourges, and S. Zuyev. Stochastic geometry and architecture of communication networks. *Telecom. Systems*, 7(1):209–227, 1997.

[4] R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(02):194–197, 1941.

[5] R. Correa, F. Havet, and J-S. Sereni. About a Brooks-type theorem for improper colouring. *Australasian Journal of Combinatorics*, 43:219–230, 2009.

[6] L. J. Cowen, R. H. Cowen, and D. R. Woodall. Defective colorings of graphs in surfaces: Partitions into subgraphs of bounded valency. *Journal of Graph Theory*, 10(2):187–195, 1986.

[7] L.J. Cowen, W. Goddard, and C.E. Jesurum. Defective coloring revisited. *J. Graph Theory*, 24:205–219, 1995.

[8] M. Fischetti, C. Lepschy, G. Minerva, G. Romanin-Jacur, and E. Toto. Frequency assignment in mobile radio systems using branch-and-cut techniques. *European Journal of Operational Research*, 123(2):241–255, 2000.

[9] P. Gupta and P.R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.

[10] M. Haenggi, J.G. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti. Stochastic geometry and random graphs for the analysis and design of wireless networks. *Selected Areas in Communications, IEEE Journal on*, 27(7):1029–1046, 2009.

[11] Frédéric Havet, Bruce Reed, and Jean-Sébastien Sereni. L(2,1)-labelling of graphs. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 621–630, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[12] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[13] L. Lovász. On decompositions of graphs. *Studia Sci. Math. Hungar.*, 1:273–238, 1966.

[14] C. Mannino and A. Sassano. An enumerative algorithm for the frequency assignment problem. *Discrete Applied Mathematics*, 129(1):155–169, 2003.

[15] D. R. Woodall. Improper colorings of graphs. In R. Nelson and R. J. Wilson, editors, *Pitman Res. Notes Math. Ser.*, volume 218, pages 45–63. Longman Scientific and Technical, 1990.

[16] Roger K. Yeh. A survey on labeling graphs with a condition at distance two. *Discrete Mathematics*, 306(12):1217 – 1231, 2006.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399

Annexe B

Distance moyenne dans les réseaux récursifs

On the average path length of deterministic and stochastics recursive networks*

Philippe J. Giabbanelli, Dorian Mazauric, and Jean-Claude Bermond

Mascotte, INRIA, I3S(CNRS,UNS), Sophia Antipolis, France,
{Philippe.Giabbanelli,Dorian.Mazauric,Jean-Claude.Bermond}@sophia.inria.fr

Abstract. The average shortest path distance ℓ between all pairs of nodes in real-world networks tends to be small compared to the number of nodes. Providing a closed-form formula for ℓ remains challenging in several network models, as shown by recent papers dedicated to this sole topic. For example, Zhang *et al.* proposed the deterministic model ZRG and studied an upper bound on ℓ. In this paper, we use graph-theoretic techniques to establish a closed-form formula for ℓ in ZRG. Our proof is of particular interests for other network models relying on similar recursive structures, as found in fractal models. We extend our approach to a stochastic version of ZRG in which layers of triangles are added with probability p. We find a first-order phase transition at the critical probability $p_c = 0.5$, from which the expected number of nodes becomes infinite whereas expected distances remain finite. We show that if triangles are added independently instead of being constrained in a layer, the first-order phase transition holds for the very same critical probability. Thus, we provide an insight showing that models can be equivalent, regardless of whether edges are added with grouping constraints. Our detailed computations also provide thorough practical cases for readers unfamiliar with graph-theoretic and probabilistic techniques.

1 Introduction

The last decade has witnessed the emergence of a new research field coined as "Network Science". Amongst well-known contributions of this field, it was found that the average distance ℓ in a myriad of real-world networks was small compared to the number of nodes (*e.g.*, in the order of the logarithm of the number of nodes). Numerous models were proposed for networks with small average distance [1, 2] such as the static Watts-Strogatz model, in which a small percentage of edges is changed in a low-dimensional lattice [3], or dynamic models in which ℓ becomes small as nodes are added to the network [4]. However, proving a closed form formula for ℓ can be a challenging task in a model, and thus this remains a current research problem with papers devoted to this sole task [5]. In this paper, we prove a closed form formula for a recently proposed model, in which the authors showed an upper bound on ℓ [6]. While the model presented

* Research funded by the EULER project and *région PACA*.

in [6] is deterministic, a stochastic version was also studied for which the authors approximated an upper bound on ℓ [8]. Thus, we present two stochastic versions and we rigorously characterize their behaviour using both upper and lower bounds on ℓ, and studying the ratio with the number of nodes.

The paper starts by establishing the notation, then each of the three Sections focusses on a model. Firstly, we consider the model as defined in [6]: we prove a closed-form formula for the average distance ℓ, and characterize the ratio between the number of nodes and ℓ. Secondly, we propose a version of the model in which edges and nodes are randomly added but in specific groups. In this version, we establish bounds on the expected value of ℓ and we provide a closed-form formula for the expected number of nodes. While the former is always finite, the latter becomes infinite from a critical probability $p_c = 0.5$, thus the ratio between ℓ and the number of nodes can be arbitrarily large. However, the infinite value of the expected number of nodes results from a few very large instances, and thus does not represent well the trend expressed by most instances for $p \geq p_c$. Consequently, we also study the ratio between the number of nodes and ℓ by considering all instances but very large ones. Thirdly, we study the number of nodes and ℓ in a stochastic version that does not impose specific groups, similarly to [8]. We show that this version also has a finite expected value for ℓ, and an infinite expected number of nodes from $p = p_c$.

2 Notation

We denote by ZRG_t the undirected graph defined by Zhang, Rong and Guo, obtained at step t [6]. It starts with ZRG_0 being a cycle of three nodes, and "ZRG_t is obtained by ZRG_{t-1} by adding for each edge created at step $t-1$ a new node and attaching it to both end nodes of the edge" [6]. The process is illustrated by Figure 1(a). We propose two probabilistic versions of ZRG. In the first one, each of the three original edges constitutes a *branch*. At each time step, a node is added *for all* active edges of a branch with independent and identical (*iid*) probability p. If a branch does not grow at a given time step, then it will not grow anymore. We denote this model by $BZRG_p$, for the probabilistic *branch* version of ZRG with probability p. Note that while the probability p is applied at each time step, the resulting graph is not limited by a number of time steps as in ZRG_t: instead, the graph grows as long as there is at least a branch for which the outcome of the stochastic process is to grow, thus there exist arbitrarily large instances. The process is illustrated in Figure 1(b). Finally, the second stochastic version differs from $BZRG_p$ by adding a node with probability p *for each* active edge. In other words, this version does not impose to grow all the 'layer' at once, but applies the probability edge by edge. We denote the last version by $EZRG_p$ for the probabilistic *edge* version of ZRG with probability p.

In this paper, we are primarily interested in the average distance. For a connected graph G having a set of nodes $V(G)$, its average distance is defined by $\ell(G) = \frac{\sum_{u \in V(G)} \sum_{v \in V(G)} d(u,v)}{|V(G)|*(|V(G)|-1)}$, where $d(u,v)$ is the length of a shortest path between u and v. In a graph with N nodes, ℓ is said to be *small* when proportional to $\ln(N)$, and *ultrasmall* when proportional to $\ln(\ln(N))$ [9].

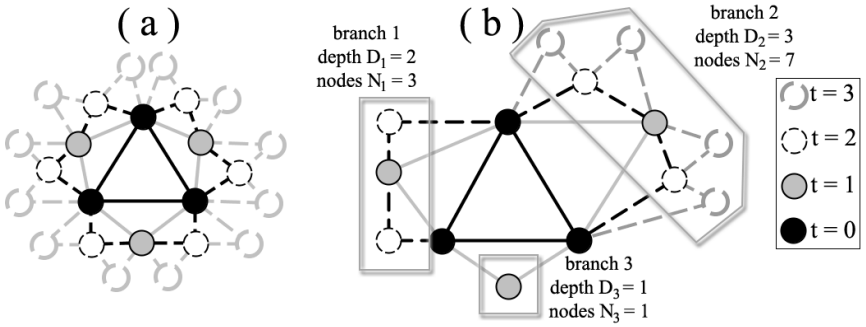


Fig. 1. The graph ZRG_0 is a triangle, or cycle of 3 nodes. At each time step, to each edge added at the previous time step, we add a node connected to the endpoints of the edge. This can be seen as adding a triangle to each outer edge. The process is depicted step by step up to ZRG_3 (a). A possible instance of $BZRG_p$ illustrates the depth and number of nodes in each of the three probabilistic branches (b). The graph grew for 3 time steps, at the end of which the outcome of the last active branch was not to grow.

3 Deterministic version

In this Section, we consider the version introduced by [6] and defined in the previous Section. We denote by V_t the vertices of ZRG_t, and A_t the vertices added at step t. We established in [7] that $|A_t| = 3 * 2^{t-1}$ for $t \geq 1$.

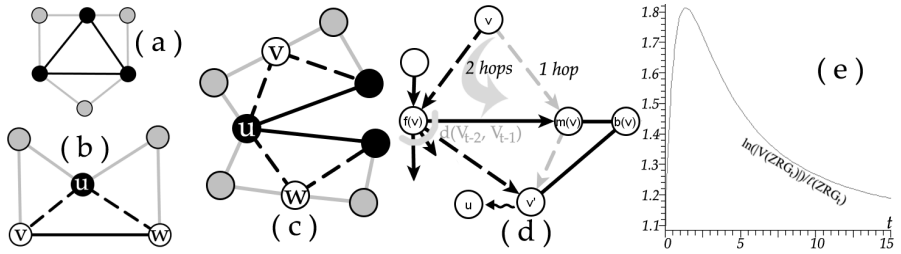


Fig. 2. In ZRG_1, each of the black initial nodes is connected to two grey added nodes (a). We assume that $u \in A_{t-1}$, thus it stems from an edge (v, w). As the edges (u, v) and (u, w) are active, u will also be connected to two nodes in A_t (b). We assume that $u \in V_{t-2}$: thus, it is connected to two (children) nodes $v, w \in A_{t-1}$. The edges (u, v) and (u, w) being active, u will also be connected to the two nodes they generate at the next step, belonging to A_t (c). Shortest paths to compute $d(A_t, V_{t-1})$ (d). The average distance in ZRG_t is close to the logarithm of the graph's size (e).

By construction, a node $u \in A_t$ is connected to the two endpoints of a formerly active edge. One endpoint was created at the step immediately before (*i.e.*, $t-1$), and we call it the *mother* $m(u) \in A_{t-1}$, while the other endpoint was created at an earlier time, and we call it the *father* $f(u) \in V_{t-2}$. A node having

same mother as u is called its uterine brother and denoted as $b(u)$. Furthermore, we observe that each node $v \in V_{t-1}$ is connected to two nodes of A_t. This is proved by induction: it holds for $t = 1$ (see Figure 2(a)), we assume it holds up to $t-1$ and we show in Figure 2(b-c) that it follows for t. Since each node in A_t is connected to two nodes in V_{t-1}, and each node in V_{t-1} is connected to two nodes in A_t, the graph has a bipartite structure used in our proof.

We now turn to the computation of $\ell(ZRG_t)$. We denote by $d(X,Y) = \sum_{u \in X} \sum_{v \in Y} d(u,v)$ the sum of distances from all nodes in X to all nodes in Y. Theorem 1 establishes the value of $g(t) = d(V_t, V_t)$, from which we will establish the average distance using $\ell(ZRG_t) = \frac{g(t)}{|V(ZRG_t)|*(|V(ZRG_t)|-1)}$. We gave the sketch of a different proof in [7], thus the interested reader can compare it with the full proof given here to better illustrate graph-theoretic techniques.

Theorem 1. $g(t) = 4^t(6t+3) + 2 * 3^t$

Proof. By definition, $V_t = V_{t-1} \cup A_t$. Thus, $d(V_t, V_t) = d(V_{t-1}, V_{t-1}) + d(A_t, V_{t-1}) + d(V_{t-1}, A_t) + d(A_t, A_t)$. Since the underlying graph is undirected, $d(A_t, V_{t-1}) = d(V_{t-1}, A_t)$ hence

$$g(t) = g(t-1) + 2d(A_t, V_{t-1}) + d(A_t, A_t), t \geq 2 \tag{1}$$

In the following, we consider that a shortest path from $v \in A_t$ always goes through $f(v)$, unless the target is the brother $b(v)$ or the mother $m(v)$ in which case correction factors are applied. Suppose that we instead go through $m(v)$ to reach some node u: since $m(v)$ is only connected to $b(v)$, $f(v)$ and some node v' (see Figure 2(d)) then the route continues through v'. However, the path $v, m(v), v'$ can be replaced by $v, f(v), v'$ without changing the length.

We compute $d(A_t, V_{t-1})$ for $t \geq 2$. Since we always go through $f(v)$, we use a path of length 2 in order to go from v to $m(v)$ whereas it takes 1 using the direct link. Thus, we have to decrease the distance by 1 for each $v \in A_t$, hence a correcting factor $-|A_t|$. We observe that each node in V_{t-2} is the father of two nodes in A_t, hence routing through the father costs $2d(V_{t-2}, V_{t-1})$ to which we add the number of times we use the edge from v to the father. As each $v \in A_t$ goes to each $w \in V_{t-1}$, the total cost is

$$d(A_t, V_{t-1}) = 2d(V_{t-2}, V_{t-1}) + |A_t||V_{t-1}| - |A_t| \tag{2}$$

We have that $2d(V_{t-2}, V_{t-1}) = 2d(V_{t-2}, V_{t-2}) + 2d(V_{t-2}, A_{t-1}) = 2g(t-2) + 2d(V_{t-2}, A_{t-1})$. Furthermore, using Eq. 1 we obtain $g(t-1) = g(t-2) + 2d(A_{t-1}, V_{t-2}) + d(A_{t-1}, A_{t-1}) \Leftrightarrow 2d(A_{t-1}, V_{t-2}) = g(t-1) - g(t-2) - d(A_{t-1}, A_{t-1})$. Substituting these equations with Eq. 2, it follows that

$$d(A_t, V_{t-1}) = g(t-1) + g(t-2) - d(A_{t-1}, A_{t-1}) + |A_t||V_{t-1}| - |A_t| \tag{3}$$

We compute $d(A_t, A_t)$, for $t \geq 2$. In order to go from v to its uterine brother $b(v) \in A_t$, it takes 2 hops through their shared mother, whereas it takes 3 hops through the father. Thus, we have a correction of 1 for $|A_t|$ nodes. The path from a v to a w is used four times, since $f(v)$ has two children in A_t and so does

$f(w)$. Finally, we add 2 for the cost of going from a node to its father at both ends of the path, and we have $|A_t|(|A_t|-1)$ such paths. Thus, the overall cost is

$$d(A_t, A_t) = 4g(t-2) + 2|A_t|(|A_t|-1) - |A_t| \quad (4)$$

We combine. Given that $|A_t| = |V_{t-1}|$, we substitute Eq. 3 into Eq. 1 hence

$$g(t) = 3g(t-1) + 2g(t-2) + d(A_t, A_t) - 2d(A_{t-1}, A_{t-1}) + 2|A_t|^2 - 2|A_t| \quad (5)$$

From Eq. 4, for $t \geq 3$ we obtain $d(A_{t-1}, A_{t-1}) = 4g(t-3)+2|A_{t-1}|^2-3|A_{t-1}|$. Given that $|A_t| = 2|A_{t-1}|$, we substitute $d(A_{t-1}, A_{t-1})$ and Eq. 4 in Eq. 5:

$$g(t) = 3g(t-1) + 6g(t-2) - 8g(t-3) + 3|A_t|^2 - 2|A_t|, t \geq 3 \quad (6)$$

We manually count that $f(0) = 6$, $f(1) = 42$ and $f(2) = 252$. Thus the equation can be solved into $g(t) = 4^t(6t+3) + 3 * 2^t$ using standard software.

Corollary 1. *Since* $|V(ZRG_t)| = 3 * 2^t$ *[6], it follows from the Theorem that* $\ell(ZRG_t) = \frac{4^t(6t+3)+3*2^t}{3*2^t(3*2^t-1)} = \frac{t2^{t+1}+2^t+1}{3*2^t-1}$.

Using this corollary, we obtain $\lim_{t\to\infty} \frac{\ln(|V(ZRG_t)|)}{\ell(ZRG_t)} = \frac{3*\ln(2)}{2} \approx 1.03$ and $\lim_{t\to\infty} \frac{\ln(\ln(|V(ZRG_t)|))}{\ell(ZRG_t)} \approx 0$. Thus, the average size is almost exactly $\ln(|V(G)|)$ for large t. Since the size of the graph is exponential in t, it is important that the graphs obtained for small values of t have a similar ratio, which is confirmed by the behaviour illustrated in Figure 2(e).

4 Stochastic branch version

As in the previous Section, we are interested in the ratio between the number of nodes and the average path length. In this Section, our approach is in three steps. Firstly, we establish bounds on the *depth* of branches, defined as the number of times that a branch has grown. Secondly, we study the number of nodes. We find that the number of nodes undergoes a first-order phase transition at the critical probability $p_c = 0.5$: for $p < 0.5$, there is a finite number of nodes, whereas for $p \geq 0.5$ this number becomes infinite. Since in the latter the expected depth of branches is bounded by finite numbers, the expected graphs have an arbitrarily small average distance compared to the number of nodes. However, the expected number of nodes only provides a mean-field behaviour that can may lack representativeness due to a few very large instances. Thus, we conclude by investigating the behavior of instances of bounded depth.

4.1 Depth of branches

To fully characterize the depth of branches, we are interested in their expected depth for the standard case as well as the two extremal cases consisting of the *deepest* and *shallowest* branches. In other words, we study the mean-field behaviour and we provide a lower and an upper bound. We start by introducing our notation for the depth in Definition 1. We start by establishing the expected depth of a branch in Theorem 2, then we turn to the expected shallowest depth, and we conclude in Theorem 4 showing that the expected deepest depth of a branch is finite.

Definition 1. *We denote D_1, D_2, D_3 the depth of the three branches. The depth of the deepest branch is $D_{max} = max(D_1, D_2, D_3)$ and the depth of the shallowest branch is $D_{min} = min(D_1, D_2, D_3)$.*

Theorem 2. *The expected depth of a branch is $\mathbb{E}(D_i) = \frac{p}{1-p}$, $i \in \{1, 2, 3\}$.*

Proof. The probability $P(D_i = k)$ that a branch grows to depth k is the probability p^k of successily growing k times, and the probability not to grow once (*i.e.*, to stop at depth $k+1$). Thus, $P(D_i = k) = \underbrace{p \cdot \cdots \cdot p}_{k} \cdot (1-p) = p^k(1-p)$. Since the expected value of a discrete random variable D_i is given by $\mathbb{E}(D_i) = \sum_{k=0}^{\infty}(kP(D_i = k))$, it follows that $\mathbb{E}(D_i) = \sum_{k=0}^{\infty}(kp^k(1-p)) = p(1-p)\sum_{k=0}^{\infty}(kp^{k-1})$. Since $\frac{dp^k}{dp} = kp^{k-1}$, we further simplify into $\mathbb{E}(D_i) = p(1-p)\sum_{k=0}^{\infty}(\frac{dp^k}{dp})$. As the sum of a derivative is equal to the derivative of the sum, it follows that $\mathbb{E}(D_i) = p(1-p)\frac{d\sum_{k=0}^{\infty}p^k}{dp}$. We note that $\sum_{k=0}^{\infty}p^k$ is an infinite geometric sum, hence

$$\mathbb{E}(D_i) = p(1-p)\frac{d}{dp}\frac{1}{1-p} = \frac{p(1-p)}{(1-p)^2} = \frac{p}{1-p}$$

Theorem 3. $\mathbb{E}(D_{min}) = -\frac{p^3}{p^3-1}$

Proof. The probability of the shallowest depth to be at least k knowing that the probability p applies iid to each branch is $P(D_{min} \geq k) = P(D_1 \geq k)P(D_2 \geq k)P(D_3 \geq k) = p^{3k}$. By definition, $P(D_{min} = k) = P(D_{min} \geq k) - P(D_{min} \geq k+1)$, thus $P(D_{min} = k) = p^{3k} - p^{3(k+1)}$. This probability is plugged into the definition of the expected value as in Theorem 2 hence

$$\mathbb{E}(D_{min}) = \sum_{k=0}^{\infty}(kP(D_{min} = k)) = \sum_{k=0}^{\infty}(k(p^{3k} - p^{3(k+1)})) = -\frac{p^3}{p^3-1}$$

Theorem 4. $\mathbb{E}(D_{max}) = -\frac{p(p^3+4p^2+3p+3)}{(p-1)(p^2+p+1)(p+1)}$.

Proof. By construction, the deepest branch does not exceed k iff none of the branches has a depth exceeding k. Since the probability p applies iid to all three branches, we have $P(D_{max} \leq k) = P(D_1 \leq k)P(D_2 \leq k)P(D_3 \leq k)$. Furthermore, a branch is strictly deeper than k if it successfully reaches depth $k+1$. Thus, $P(D_i > k) = \underbrace{p \cdot \cdots \cdot p}_{k+1} = p^{k+1}$, $i \in \{1, 2, 3\}$. By algebraic simplification, we have $P(D_{max} \leq k) = (1 - P(D_1 > k))(1 - P(D_2 > k))(1 - P(D_3 > k)) = (1 - p^{k+1})^3$. By definition, $P(D_{max} = k) = P(D \leq k) - P(D \leq k-1) = (1 - p^{k+1})^3 - (1 - p^k)^3$. Given that $\mathbb{E}(D_{max}) = \sum_{k=0}^{\infty}(kP(D_{max} = k))$, we replace the expression of $P(D_{max} = k)$ to obtain $\mathbb{E}(D_{max}) = \sum_{k=0}^{\infty}(k((1 - p^{k+1})^3 - (1 - p^k)^3))$. The final expression results from algebraic simplification using standard software.

4.2 Average number of nodes

We introduce our notation in Definition 2, and Theorem 5 provides a closed-form of the expected number of nodes.

Definition 2. *We denote by N_1, N_2, and N_3 the number of nodes in the three branches. Since we start from a cycle with three nodes, the total number of nodes is $N = N_1 + N_2 + N_3 + 3$.*

Theorem 5. *For $p < \frac{1}{2}$, the expected number of nodes is $\mathbb{E}(N) = \frac{3(1-p)}{1-2p}$.*

Proof. First, we focus on the expected number of nodes in a branch. As the probability p applies iid to all three branches, we select the first branch without loss of generality. By construction, the total number of nodes N_1 in the branch 1 at depth $D_1 = k \geq 0$ is $N_1 = 2^k - 1 = \sum_{i=1}^{k} 2^{i-1}$. Thus, the expected value of the random variable N_1 is given by $\mathbb{E}(N_1) = \sum_{k=0}^{\infty}((2^k - 1)P(D_1 = k))$. As shown in Theorem 2, $P(D_1 = k) = p^k(1-p)$. We replace it in the equation to obtain $\mathbb{E}(N_1) = \sum_{k=0}^{\infty}((2^k - 1)p^k(1-p))$. After expanding the equation, we have $\mathbb{E}(N_1) = \sum_{k=0}^{\infty}(2^k p^k(1-p) - p^k(1-p)) = (1-p)\sum_{k=0}^{\infty}((2p)^k) - (1-p)\sum_{k=0}^{\infty}(p^k)$. As in Theorem 2, we have $\sum_{k=0}^{\infty}(p^k) = \frac{1}{1-p}$ thus the second term simplifies and yields $\mathbb{E}(N_1) = (1-p)\sum_{k=0}^{\infty}((2p)^k) - 1$. It is well known that a serie of the type $\sum_{k=0}^{\infty}(x^k)$ diverges to infinity for $x \geq 1$. Thus, our serie diverges for $2p \geq 1 \Leftrightarrow p \geq \frac{1}{2}$. In other words, this results only holds for $0 \leq p < \frac{1}{2}$.
The infinite geometric sum $\sum_{k=0}^{\infty}((2p)^k)$ can be simplified in $\frac{1}{1-2p}$ hence $\mathbb{E}(N_1) = \frac{1-p}{1-2p} - 1 = \frac{1-p-1+2p}{1-2p} = \frac{p}{1-2p}$. The probability p applies iid to all three branches hence $\mathbb{E}(N_1) = \mathbb{E}(N_2) = \mathbb{E}(N_3)$. Thus, by Definition 2, the expected number of nodes in the overall graph is given by $\mathbb{E}(N) = 3 + 3\mathbb{E}(N_1) = 3 + \frac{3p}{1-2p} = \frac{3(1-p)}{1-2p}$.

Theorem 5 proved that the average number of nodes diverges to infinity at the critical probability $p_c = 0.5$. This may appear to be a discrepancy with Theorem 4 stating that the expected depth of the deepest branch is finitely bounded. For the sake of clarity, we provide an intuition and an example on this point. First, we note that the *expected* deepest depth and the *expected* number of nodes have different growth rates. Indeed, even if graphs with very deep depth scarcely occur at $p = 0.5$, their impact on the expected number of nodes is tremendous since the number of nodes grows *exponentially* with the depth. On the other hand, the impact of such graphs on the expected deepest depth is only linear. To illustrate different growth rates with a known network, consider the complete graph K_n, in which each of the $n \geq 1$ nodes is connected to all others. In K_n, the number of nodes grows linearly whereas the distance is constant. Thus, the distance between two nodes is 1 even with an infinite number of nodes.

In a nutshell, the expected number of nodes for $p \geq 0.5$ may not be representative of most instances due to the large impact of very deep graphs. Thus, it remains of interest to investigate the number of nodes for graphs with bounded depths. This is established in Theorem 6, in which we consider the $q\%$ possible instances with smallest depth. By lowering the impact of the very deep graphs, this theorem consistutes a lower bound that better describes practical cases.

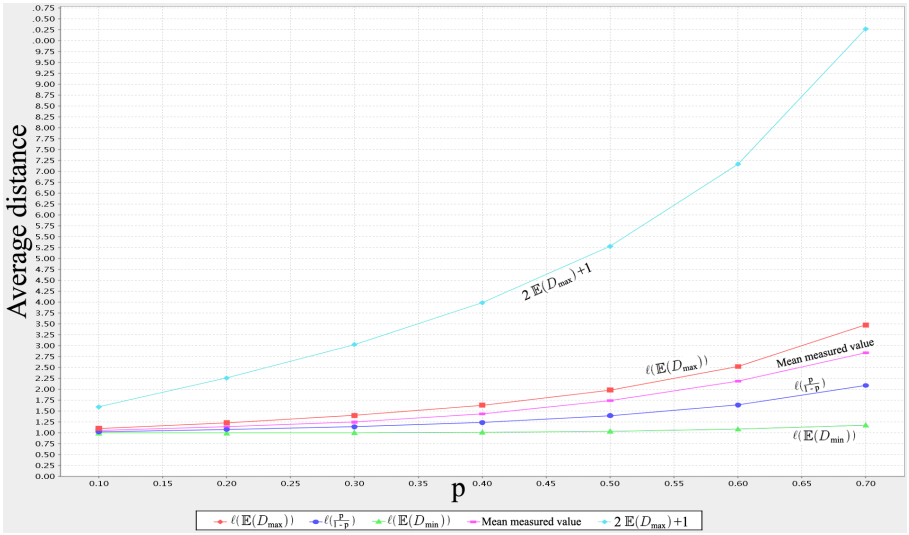


Fig. 3. The measured average distance compared with three bounds and an estimate, proved or conjectures. The simulations valide the conjectures. *Color online.*

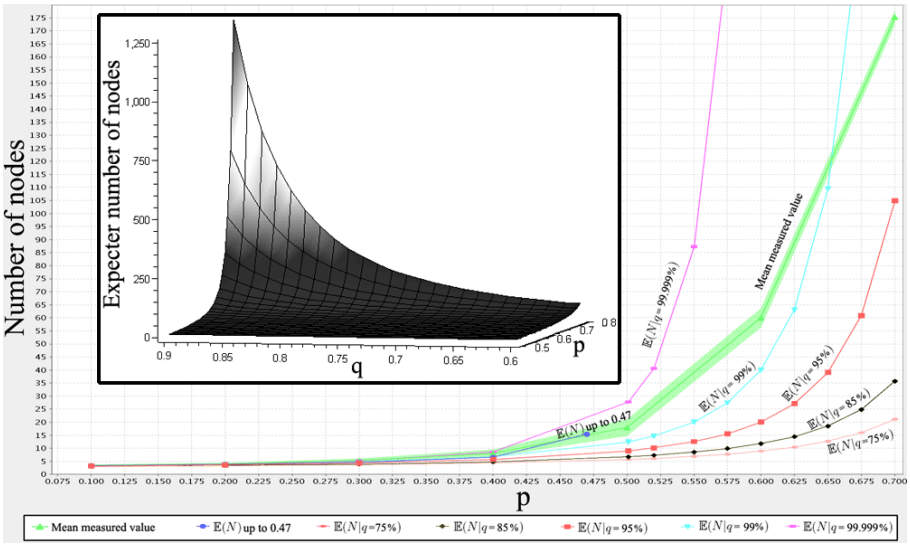


Fig. 4. Up to $p = 0.5$ (excluded), the expected number of nodes is finite. From $p = 0.5$, the expected number of nodes is infinite due to very large instances, thus we provide a finite estimate by considering the $q\%$ smallest instances, from $q = 75\%$ to $q = 100 - 10^{-3}\%$. Simulations were limited to the graphs that fitted in memory thus the mean measured value represents only an estimate based on small graphs for values beyond $p = 0.5$. *Color online.*

Lemma 1. $\mathbb{E}(N|D_{max} \leq K) = -\frac{(3p-6p^{K+2}2^K+6p^{K+1}2^K-3)}{(2p-1)(p^{K+1}-1}$

Proof. The expected number of nodes is adapted from Theorem 5 by the following substitutions: $\mathbb{E}\underbrace{(N_1)}_{(N_1|D_{max}\leq K)} = \sum_{k=0}^{\infty}((2^k-1)\underbrace{P(D_1=k)}_{P(D_1=k|D_{max}\leq K)})$. We simplify:

$$P(D_1 = k|D_{max} \leq K) \overset{\text{formula}}{=} \frac{P(D_1=k\cap D_{max}\leq K)}{P(D_{max}\leq K)}$$

$$\overset{\text{expend } D_{max}}{=} \frac{P(D_1=k\cap D_1\leq K\cap D_2\leq K\cap D_3\leq K)}{P(D_1\leq K\cap D_2\leq K\cap D_3\leq K)}$$

$$\overset{\text{independence}}{=} \frac{P(D_1=k)P(D_1\leq K)P(D_2\leq K)P(D_3\leq K)}{P(D_1\leq K)P(D_2\leq K)P(D_3\leq K)} \overset{\text{simplifying}}{=} \frac{P(D_1=k)}{P(D_1\leq K)}$$

Thus $\mathbb{E}(N_1|D_{max} \leq K) = \sum_{k=0}^{\infty}((2^k-1)\frac{P(D_1=k)}{P(D_1\leq K)})$. We showed in the proof of Theorem 4 that $P(D_1 \leq K) = 1-p^{K+1}$, and we showed in the proof of Theorem 2 that $P(D_1 = k) = p^k(1-p)$. By substituting these results, and using from the previous Theorem that $\mathbb{E}(N) = 3 + 3\mathbb{E}(N_1)$, it follows that

$$\mathbb{E}(N|D_{max} \leq K) = 3 + 3\sum_{k=0}^{K}(\frac{(2^k-1)p^k(1-p)}{1-p^{K+1}})$$

The closed form formula follows by algebraic simplification.

Theorem 6. *By abuse of notation, we denote by $\mathbb{E}(N|q)$ the expected number of nodes for the q% of instances of $BZRG_p$ with smallest depth. We have*

$$\mathbb{E}(N|q) = -\frac{3(-1+(1-\sqrt[3]{q})^{\frac{\ln(2)+\ln(p)}{\ln(p)}})(p-1)}{-\sqrt[3]{q}(2p-1)}$$

Proof. Theorem 4 proved that the expected deepest depth of a branch was at most K with probability $(1-p^{K+1})^3$. Thus, if we want this probability to be q, we have to consider branches whose depth K is at most:

$$(1-p^{K+1})^3 = q \Leftrightarrow K = log_p(1-\sqrt[3]{q}) - 1$$

The Theorem follows by replacing K in Lemma 1 with the value above.

The effect of a few percentages of difference in q is shown in Figure 4 together with the results from Theorem 5 and 6. In the inset of Figure 4, we show that the number of nodes grows sharply with q.

4.3 Average path length

We conducted experiments in order to ensure the veracity of the results presented in the two previous Sections, and to compare them with devised bounds. For values of p from 0.1 to 0.7 by steps of 0.1, we measured the average distance of the resulting graphs, obtained as the average over 1000 instances. In Figure 3, we plot it against four bounds and an estimated mean:

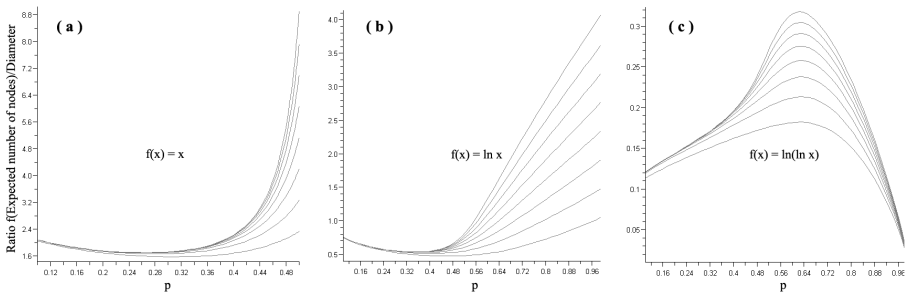


Fig. 5. We measured the ratio between $\mathbb{E}(N|100-10^x)$ (a), $\ln(\mathbb{E}(N|100-10^x))$ (b), $\ln(\ln(\mathbb{E}(N|100-10^x)))$ (c) and the diameter $2\mathbb{E}(D_{max})+1$ for x going from 0 (bottom curve) to 7 (top curve). We determined a critical probability $p_c = 0.5$ at which the regime changes, and this is confirmed by these measures showing that the average distance goes from linear in the number of nodes (a) for $p < p_c$ to small in the number of nodes (b) for $p \geq p_c$.

- *Proven bound.* Theorem 4 established the expected deepest depth. At any time, the graph has three branches, and we can only go from one branch to another through the basic cycle. Thus, the expected maximum distance between any two points in the graph consists of going from the most remote node of two branches to the cycle, and going through the cycle. As a node is at most at distance $\mathbb{E}(D_{max})$ from the cycle and we can go from any node of the cycle to another in one hop, the expected maximum distance is $2\mathbb{E}(D_{max}) + 1$. Since this is the *maximum* distance, we use it as a proven upper bound on the *average* distance.
- *Conjecture bounds.* Our intuition is that since $\ell(t)$, proven in Theorem 1, provides the average distance for a graph in which *all* branches have depth t, then a lower and upper bound can be obtained by considering the graphs with shallowest (Theorem 3) and deepest (Theorem 4) depths respectively. This is confirmed by the simulations.
- *Conjectured mean.* Similarly to the conjecture bounds, we have proven the expected depth $\mathbb{E}(D) = \frac{p}{1-p}$ of a branch in Theorem 2, and the simulation confirms that $\ell(\frac{p}{1-p})$ constitute an estimate of the average distance.

As we previously did for the deterministic version, we now investigate whether the average distance $\ell(BZRG_p)$ can be deemed small compared to the number of nodes $|V|$. As explained in the previous Section, we proved a (first-order) phase transition at the critical probability $p_c = 0.5$. The behaviour of the graph can be characterized using the ratios displayed in Figure 5: for $p \ll p_c$, we observe an average distance proportional to the number of nodes, and for $p > p_c - \epsilon$ the average distance is proportional to the logarithm of the number of nodes which is deemed small. The ratio in Figure 5(c) is too low, and tends to 0 for a large probability p, thus the graph cannot be considered ultra-small. The separation at $p_c - \epsilon$ can also be understood from a theoretical perspective. For $p \geq p_c$, we proved that $\ell(BZRG_p)$ can be arbitrary small compared to $|V|$ since $\ell(BZRG_p)$

is finite whereas $|V|$ is infinite. When $p = 0.5-\varepsilon$, the average distance is bounded by a finite number: by Theorem 2 we have that the expected depth of a branch is $\mathbb{E}(D_i) < 1$ and, using the aforementioned argument regarding the maximum distance, this entails $\ell(BZRG_p) < 2 * 1 + 1 = 3$. Furthermore, as stated in the proof of Theorem 6, the expected number of nodes in a branch is $\mathbb{E}(N_i) = \frac{0.5-\varepsilon}{\varepsilon}$ which can thus be arbitrarily large. Thus, the behaviour for $p \geq p_c$ is also expected to hold in a small neighborhood of the critical probability.

5 Stochastic edge version

In order to show a broad variety of approaches, we prove the number of nodes and the average path length of $EZRG_p$ using different tools from the previous Section. Theorem 7 establishes the number of nodes in the graph.

Theorem 7. *For $p < \frac{1}{2}$, the expected number of nodes is $\mathbb{E}(N) = 3 + \frac{3p}{1-2p}$.*

Proof. We consider the dual of the graph, which we define using a visual example in Figure 6. The dual is a binary tree: for an edge, a triangle is added (root of the tree) with probability p, to which two triangles are added iid at the next step (left and right branches of the tree) with probability p, and so on. Since one node is added to the tree when a node is added to the original graph, studying the number of nodes in $EZRG_p$ is equivalent to studying the number of nodes in the **t**ree. We denote the latter by $t(p)$. The number of nodes starting from any edge is the same, and we denote it by N. Thus, N corresponds to the probability of starting a tree (*i.e.*, adding a first node that will be the root) multipled by the number of nodes in the tree: $N = pt(p)$. Once the tree has been started, the number of nodes corresponds to the sum of the root and the number of nodes in the two branches, hence $t(p) = 2pt(p) + 1$. Note that there is a solution if and only if $p < \frac{1}{2}$, and otherwise the number of nodes is infinite. By arithmetic simplification, we obtain $t(p)(1-2p) = 1 \Leftrightarrow t(p) = \frac{1}{1-2p}$. Thus, $N = \frac{p}{1-2p}$. Since the graph starts as a cycle of length three, the number of counts corresponds to the three original nodes, to which we add the number of nodes starting in each of the three trees, thus $\mathbb{E}(N) = 3 + \frac{3p}{1-2p} = \frac{3(1-p)}{1-2p}$.

A proof similar to Theorem 7 could be applied to the number of nodes in $BZRG_p$. However, the tree associated with $BZRG_p$ grows by a complete level with probability p, instead of one node at each time. Thus, the current depth of the tree has to be known by the function in order to add the corresponding number of nodes, hence $N = pt(p, 0)$ and $t(p, k) = pt(p, k + 1) * 2^k$.

In the previous model, we showed that the expected average distance had a constant growth whereas the expected number of nodes had an exponential growth. Thus, the gap between the two could be arbitrarily large. Using simulations reported in Figure 7, we show that the same effect holds in this model.

6 Conclusion and future work

We proved a close-form formula for the average distance in ZRG_t. We proposed two stochastic versions, and showed that they had a first-order phase transition

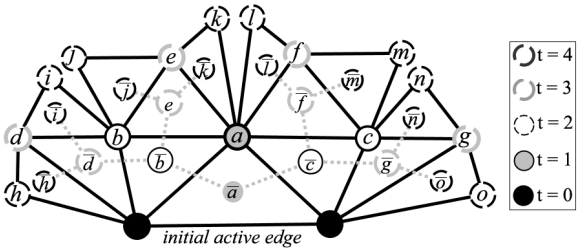


Fig. 6. Nodes linked by black edges correspond to four successive growths from an initial active edge. When a node x is added, it creates a triangle, to which we associate a node $\bar{x}$. If two triangles share an edge, their nodes $\bar{x_1}$ and $\bar{x_2}$ are connected by a grey dashed edge. The graph formed by the nodes associated to triangles is called *dual*.

at the same critical probability. In the recent years, we have witnessed many complex network models in which nodes are added at each time step. The graph-theoretic and probabilistic techniques illustrated in our paper can thus be used to rigorously prove the behaviour of models.

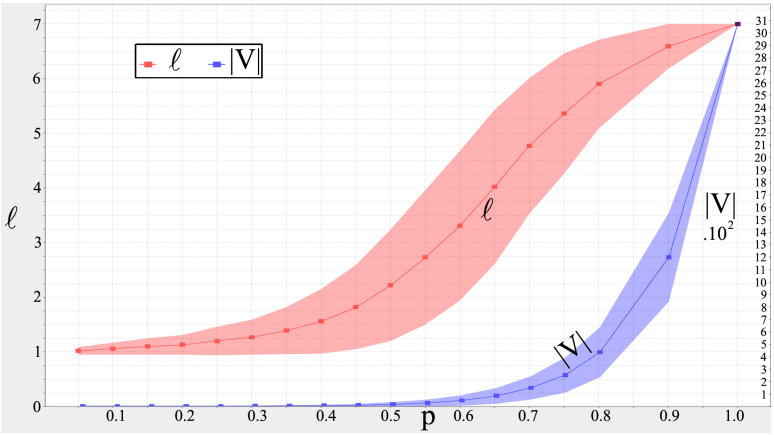


Fig. 7. The average path length has a slow growth compared to the number of nodes, as in $BZRG_p$. We show the values averaged accross simulations and the standard deviation. *Color online.*

References

1. S. Schnettler, Social Networks **31**, 165 (July 2009), ISSN 03788733
2. P. Giabbanelli and J. Peters, Technique et Science Informatiques (to appear)(2010)
3. D. J. Watts, *Small worlds: the dynamics of networks between order and randomness* (Princeton University Press, Princeton, NJ, 1999)
4. J. Davidsen, H. Ebel, and S. Bornholdt, Phys. Rev. Lett. **88** (2002)
5. Z. Zhang, L. Chen, S. Zhou, L. Fang, J. Guan, and T. Zou, Phys. Rev. E **77** (2008)
6. Z. Zhang, L. Rong, and C. Guo, Physica A **363**, 567 (2006)
7. P. Giabbanelli, D. Mazauric, and S. Perennes, in *Proc. of the 12th AlgoTel*, 2010
8. Z. Zhang, L. Rong, and F. Comellas, J. of Physics A **39**, 3253 (2006)
9. R. Cohen and S. Havlin, Phys. Rev. Lett. **90** (2003)

Bibliographie

[ABG+11a] J. Araujo, J-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski. Weighted improper colouring. Technical Report RR-7590, INRIA, April 2011. (Cité en page 6.)

[ABG+11b] J. Araujo, J-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski. Weighted improper colouring. In *International Workshop on Combinatorial Algorithms (IWOCA'11)*, page 12p, Victoria, Canada, 2011. (Cité en page 6.)

[Ack79] M. Ackroyd. Call repacking in connecting networks. *IEEE Transactions on Communications*, 27(3) :589–591, March 1979. (Cité en page 13.)

[AWW05] I.F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks : a survey. *Computer Networks*, 47(4) :445 – 487, 2005. (Cité en page 95.)

[BBJMR09a] A-E. Baert, V. Boudet, A. Jean-Marie, and X. Roche. Minimization of download time variance in a distributed vod system. *Scalable Computing Practice and experience*, 10(1) :75–86, 2009. (Cité en page 134.)

[BBJMR09b] A-E. Baert, V. Boudet, A. Jean-Marie, and X. Roche. performance analysis of dat replication in grid delivery networks. In *Int. Conf. on Complex Intellignet and Software Intensive Systems*, pages 369–374, 2009. (Cité en pages 5 et 134.)

[BCL+03] M. Bouklit, D. Coudert, J-F. Lalande, C. Paul, and H. Rivano. Approximate multicommodity flow for WDM networks design. In J. Sibeyn, editor, *10th International Colloquium on Structural Information and Communication Complexity – SIROCCO*, number 17 in Proceedings in Informatics, pages 43–56, Umea, Sweden, 2003. Carleton Scientific. (Cité en page 12.)

[BCLR03] M. Bouklit, D. Coudert, J-F. Lalande, and H. Rivano. Approximation combinatoire de multiflot fractionnaire : améliorations. In *5ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'03)*, Banyuls-sur-mer, France, May 2003. (Cité en page 12.)

[BCM+11] Sonia Belhareth, David Coudert, Dorian Mazauric, Nicolas Nisse, and Issam Tahiri. Reconfiguration avec contraintes physiques dans les réseaux WDM. In Ducourthial et Bertrand et Felber et Pascal, editor, *13es Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel)*, Cap Estérel, France, 2011. (Cité en pages 29 et 75.)

[BEP+96] K. Bala, G. Ellinas, M. Post, Chien-Chung Shen, J. Wei, and N. Antoniades. Towards hitless reconfiguration in WDM optical networks for ATM transport. In *IEEE Global Telecommunications Conference (Globecom)*, volume 1, pages 316–320 vol.1, Nov 1996. (Cité en page 13.)

[BF02] H. L. Bodlaender and F. V. Fomin. Approximation of pathwidth of outerplanar graphs. *J. Algorithms*, 43(2) :190–200, 2002. (Cité en page 49.)

[BFFS02] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *14th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 200–209, San Diego, California, USA (part of FCRC 2003), June 2002. ACM. (Cité en page 25.)

[BFST03] L. Barrière, P. Fraigniaud, N. Santoro, and D. M. Thilikos. Searching is not jumping. In Hans L. Bodlaender, editor, *29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2880 of *Lecture Notes in Computer Science*, pages 34–45, Elspeet, The Netherlands, June 2003. Springer. (Cité en page 25.)

[BGM10] J-C. Bermond, P. Giabbanelli, and D. Mazauric. Average path length of deterministic and stochastics recursive networks. In *Second Workshop on Complex Networks (CompleNet'10)*, page 12p, Rio de Janeiro, Brazil, 2010. (Cité en page 7.)

[BJMMY11] J-C. Bermond, A. Jean-Marie, D. Mazauric, and J. Yu. Well balanced designs for data placement. Technical Report RR-7725, INRIA, September 2011. (Cité en pages 5, 133 et 148.)

[BK96] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21 :358–402, September 1996. (Cité en pages 49 et 76.)

[BKP03] S. Beker, D. Kofman, and N. Puech. Off-line MPLS layout design and reconfiguration : Reducing complexity under dynamic traffic conditions. In *International Network Optimization Conference (INOC)*, pages 61–66, October 2003. (Cité en page 13.)

[BM00] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks : Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking*, 8(5) :598–607, October 2000. (Cité en page 13.)

[BMMN08] J-C. Bermond, D. Mazauric, V. Misra, and P. Nain. Distributed call scheduling in wireless networks. Technical Report RR-6763, INRIA, December 2008. (Cité en pages 5 et 107.)

[BMMN09] J-C. Bermond, D. Mazauric, V. Misra, and P. Nain. Algorithmes distribués d'ordonnancement dans les réseaux sans-fil. In *10èmes Journées Doctorales en Informatique et Réseaux (JDIR)*, Belfort, France, February 2009. (Cité en pages 5 et 107.)

[BMMN10] J-C. Bermond, D. Mazauric, V. Misra, and P. Nain. A distributed scheduling algorithm for wireless networks with constant overhead and arbitrary binary interference. In *SIGMETRICS 2010*. ACM, 2010. 2-page paper. to appear. (Cité en pages 5 et 107.)

[BPF04] S. Beker, N. Puech, and V. Friderikos. A tabu search heuristic for the off-line MPLS reduced complexity layout design problem. In *3rd International IFIP-TC6 Networking Conference (Networking)*, volume 3042 of *Lecture Notes in Computer Science*, pages 514–525, Athens, Greece, May 2004. Springer. (Cité en page 13.)

[Bre67] R. L. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers*, VI(5) :72–78, 1967. (Cité en page 24.)

[BSS09] L. X. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Transactions on Networking*, 17(5) :1467–1480, 2009. (Cité en pages 4, 5, 108, 110, 112, 113, 114, 127, 128 et 129.)

[BV09] Vartika Bhandari and Nitin H. Vaidya. A result on hybrid scheduling in wireless networks. Technical report, University of Illinois, Dept. Electrical and Computer Eng., March 2009. (Cité en page 125.)

[BZM06] Andrew Brzezinski, Gil Zussman, and Eytan Modiano. Enabling distributed throughput maximization in wireless mesh networks : a partitioning approach. In *MobiCom*, pages 26–37. ACM, 2006. (Cité en page 112.)

[Cam89] Kathie Cameron. Induced matchings. *Discrete Applied Mathematics*, 24(1-3) :97–102, 1989. (Cité en page 111.)

[CBL07] Xiaowen Chu, Tianming Bu, and Xiang-Yang Li. A study of lightpath rerouting schemes in wavelength-routed wdm networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 2400–2405, 2007. (Cité en page 13.)

[CCM+09] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs when optimizing lightpaths reconfiguration in WDM networks. RR 7047, INRIA, 2009. (Cité en pages 4 et 11.)

[CCM+10a] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs in process strategy games with application in the WDM reconfiguration problem. In P. Boldi and L. Gargano, editors, *Fifth International conference on Fun with Algorithms (FUN 2010)*, volume 6099 of *Lecture Notes in Computer Science*, pages 121–132, Ischia Island, Italy, June 2010. Springer. (Cité en pages 4 et 11.)

[CCM+10b] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs when optimizing Lightpaths Reconfiguration in WDM networks. In *FUN*, volume 6099 of *LNCS*, pages 121–132, 2010. (Cité en pages 4 et 11.)

[CCM+ar] N. Cohen, D. Coudert, D. Mazauric, N. Nepomuceno, and N. Nisse. Tradeoffs in process strategy games with application in the WDM reconfiguration problem. *Theoretical Computer Science (TCS)*, 2011, to appear. (Cité en pages 4, 11, 58 et 60.)

[CDH+06] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, and Robert Morris. Efficient replica maintenance for distributed storage systems. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation (NSDI'06)*, pages 45–58, Berkeley, CA, USA, 2006. USENIX Association. (Cité en page 134.)

[CGM+10a] S. Caron, F. Giroire, D. Mazauric, J. Monteiro, and S. Pérennes. Data life time for different placement policies in p2p storage systems. In *Third International Conference on Data Management in Grid and P2P Systems (GLOBE 2010)*, page 12p, Bilbao, Spain, 2010. (Cité en pages 5, 133 et 135.)

[CGM+10b] S. Caron, F. Giroire, D. Mazauric, J. Monteiro, and S. Pérennes. P2p storage systems : Data life time for different placement policies. Technical Report RR-7209, INRIA, February 2010. (Cité en pages 5, 133 et 134.)

[CGM+10c] S. Caron, F. Giroire, D. Mazauric, J. Monteiro, and S. PÃÂ©rennes. P2p storage systems : Data life time for different placement policies. In Maria Gradinariu Potop-Butucaru et Hervé Rivano, editor, *12èmes Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel)*, page 4p, Belle Dune, France, 2010. (Cité en pages 5 et 133.)

[CHM08a] D. Coudert, F. Huc, and D. Mazauric. A distributed algorithm for computing and updating the process number of a forest. Research Report 6560, INRIA, 06 2008. (Cité en pages 4 et 11.)

[CHM08b] D. Coudert, F. Huc, and D. Mazauric. A distributed algorithm for computing and updating the process number of a forest. In G. Taubenfeld, editor, *22nd International Symposium on Distributed Computing (DISC)*, volume 5218 of *Lecture Notes in Computer Science*, pages 500–501, Arcachon, France, September 2008. Springer. brief announcement. (Cité en pages 4 et 11.)

[CHM+09a] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J-S. Sereni. Reconfiguration of the routing in WDM networks with two classes of services. In *13th Conference on Optical Network Design and Modeling (ONDM)*, pages 1–6, Braunschweig, Germany, February 2009. IEEE. (Cité en pages 4, 11, 13, 29 et 76.)

[CHM+09b] D. Coudert, F. Huc, D. Mazauric, N. Nisse, and J-S. Sereni. Reconfiguration of the routing in WDM networks with two classes of services. In *13th Conference on Optical Network Design and Modeling (ONDM)*, Braunschweig, Germany, February 2009. IEEE. (Cité en pages 4 et 11.)

[CHM11] D. Coudert, F. Huc, and D. Mazauric. A distributed algorithm for computing and updating the process number of a forest. *Algorithmica*, 2011. (Cité en pages 4, 11, 30, 47 et 49.)

[CHS07] D. Coudert, F. Huc, and J.-S. Sereni. Pathwidth of outerplanar graphs. *Journal of Graph Theory*, 55(1) :27–41, May 2007. (Cité en page 49.)

[CJ06] Colbourn C.J. and Dinitz J.H., editors. *The CRC Handbook of Combinatorial Designs (2nd edition)*, volume 42. CRC Press, 2006. (Cité en page 135.)

[CKH06] *On the Vertex Separation of Unicyclic Graphs*, Chang-Hua, Taiwan, April 2006. (Cité en page 49.)

[CKLS08] Prasanna Chaporkar, Koushik Kar, Xiang Luo, and Saswati Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory*, 54(2) :572–594, 2008. (Cité en page 124.)

[CLBG00] O. Crochat, J.Y. Le Boudec, and O. Gerstel. Protection interoperability for wdm optical networks. *IEEE/ACM Transactions on Networking (TON)*, 8(3) :384–395, 2000. (Cité en page 78.)

[CLM09] L. Chiaraviglio, E. Leonardi, and M. Mellia. How much can Internet be greened ? In *GreenComm*, 2009. (Cité en page 78.)

[CM08] D. Coudert and D. Mazauric. Network reconfiguration using cops-and-robber games. Technical Report inria-00315568, INRIA, August 2008. (Cité en pages 4 et 11.)

[CMN09a] L. Chiaraviglio, M. Mellia, and F. Neri. Reducing power consumption in backbone networks. In *ICC*, 2009. (Cité en page 78.)

[CMN09b] D. Coudert, D. Mazauric, and N. Nisse. On rerouting connection requests in networks with shared bandwidth. In *DIMAP workshop on Algorithmic Graph Theory (AGT09)*, volume 32 of *Electronic Notes in Discrete Mathematics*, pages 109–116, Warwick, UK, March 2009. Elsevier. (Cité en pages 4, 11 et 66.)

[CMN09c] D. Coudert, D. Mazauric, and N. Nisse. Routing reconfiguration/process number : Networks with shared bandwidth. Research Report 6790, INRIA, 01 2009. (Cité en pages 4 et 11.)

[CNR10] D. Coudert, N. Nepomuceno, and H. Rivano. Power-efficient radio configuration in fixed broadband wireless networks. *Computer Communications*, 2010. To appear. (Cité en page 78.)

[Cou10] D. Coudert. *Algorithmique et optimisation dans les réseaux de télécommunications*. Habilitation à diriger des recherches, Université de Nice Sophia-Antipolis (UNS), March 2010. (Cité en pages 12, 61 et 76.)

[CPPS05] D. Coudert, S. Perennes, Q-C. Pham, and J-S. Sereni. Rerouting requests in WDM networks. In *7èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'05)*, pages 17–20, Presqu'île de Giens, France, May 2005. (Cité en pages 21, 22, 25, 27 et 31.)

[CR02a] D. Coudert and H. Rivano. Lightpath assignment for multifibers wdm networks with wavelength translators. In *IEEE Global Telecommunications Conference – GLOBECOM*, volume 3, pages 2686–2690, November 2002. OPNT-01-5. (Cité en page 12.)

[CR02b] D. Coudert and H. Rivano. Routage optique dans les réseaux WDM multifibres avec conversion partielle. In *4ème Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel'02)*, pages 17–24, Mèze, France, May 2002. (Cité en page 12.)

[CRR03] D. Coudert, H. Rivano, and X. Roche. A combinatorial approximation algorithm for the multicommodity flow problem. In K. Jansen and R. Solis-Oba, editors, *WAOA 03*, number 2909 in Lecture Notes in Computer Science, pages 256–259, Budapest, Hungary, September 2003. Springer-Verlag. (Cité en page 12.)

[CS07] D. Coudert and J-S. Sereni. Characterization of graphs and digraphs with small process number. Research Report 6285, INRIA, September 2007. (Cité en page 52.)

[CS11] D. Coudert and J-S. Sereni. Characterization of graphs and digraphs with small process number. *Discrete Applied Mathematics (DAM)*, 159(11) :1094–1109, July 2011. (Cité en pages 23 et 31.)

[CSB+08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *IEEE INFOCOM*, pages 457–465, 2008. (Cité en page 78.)

[CXW09] Haining Chen, Xiaojuan Xie, and Hongyi Wu. A queue-aware scheduling algorithm for multihop relay wireless cellular networks. *Mobile WiMAX Symposium, IEEE*, 0 :63–68, 2009. (Cité en page 108.)

[Dan63] G.B. Dantzig. *Linear programming and extensions*. Princeton Univ Pr, 1963. (Cité en page 80.)

[Die97] R. Diestel. Graph Theory (Graduate Texts in Mathematics), 1997. (Cité en page 124.)

[DKL87] N. Deo, M.S. Krishnamoorthy, and M.A. Langston. Exact and approximate solutions for the gate matrix layout problem. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(1) :79 – 84, january 1987. (Cité en pages 22 et 23.)

[DLS+04] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. Designing a DHT for low latency and high throughput. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design*

and Implementation (NSDI '04), San Francisco, California, March 2004. (Cité en page 134.)

[DPS02] J. Díaz, J. Petit, and M. Serna. A survey on graph layout problems. *ACM Computing Surveys*, 34(3) :313–356, 2002. (Cité en pages 22 et 24.)

[DS04] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162 :2005, 2004. (Cité en page 23.)

[EAM07] A. Eryilmaz, O. Asuman, and E. Modiano. Polynomial complexity algorithms for full utilization of multi-hop wireless networks. *INFOCOM*, pages 499–507, 2007. (Cité en page 112.)

[EIS75] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science, 1975.*, pages 184–193, 1975. (Cité en page 79.)

[EM04] J.A. Ellis and M. Markov. Computing the vertex separation of unicyclic graphs. *Information and Computation*, 192(2) :123–161, 2004. (Cité en page 49.)

[EST94] J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113(1) :50–79, 1994. (Cité en page 29.)

[FT08] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3) :236–245, 2008. (Cité en page 24.)

[FW77] S. Fiorini and R. J. Wilson. *Edge-colourings of graphs*. Pitman, 1977. (Cité en page 124.)

[GJ77] M. R. Garey and D. S. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4) :826–834, 1977. (Cité en page 23.)

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. (Cité en pages 23 et 91.)

[GJS76] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1 :117–129, 1976. (Cité en pages 88 et 89.)

[GKM99] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1) :15–23, 1999. (Cité en page 80.)

[GLS07] Abhinav Gupta, Xiaojun Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. In *INFOCOM*, pages 1631–1639, 2007. (Cité en pages 4, 5, 108, 112 et 114.)

[GM03] A. Gencata and B. Mukherjee. Virtual-topology adaptation for WDM mesh networks under dynamic traffic. *IEEE/ACM Transactions on Networking*, 11(2) :236–247, April 2003. (Cité en page 13.)

[GMM11] F. Giroire, D. Mazauric, and J. Moulierac. Routage efficace en énergie. In Ducourthial et Bertrand et Felber et Pascal, editor, *13es Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel)*, Cap Estérel, France, 2011. (Cité en pages 4 et 77.)

[GMMO10a] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing Routing Energy Consumption : from theoretical to practical results. Research Report inria-00464318, INRIA, May 2010. (Cité en pages 4, 77 et 105.)

[GMMO10b] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing routing energy consumption : from theoretical to practical results. In *International Conference on Green Computing and Communications (GreenCom'10)*, page 8p, Hangzhou, China, 2010. (Cité en pages 4, 77, 94 et 95.)

[GMP09] F. Giroire, J. Monteiro, and S. Pérennes. P2P storage systems : How much locality can they tolerate ? In *Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN)*, pages 320–323, Oct 2009. (Cité en pages 133 et 134.)

[GMP10] P. Giabbanelli, D. Mazauric, and S. Pérennes. Computing the average path length and a label-based routing in a small-world graph. In *12èmes Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (AlgoTel'10)*, page 4p, Belle Dune France, 2010. (Cité en page 7.)

[Gol91] P. A. Golovach. Search number, node search number, and vertex separator of a graph. *Vestnik Leningrad University, Mathematics*, 24(1) :88–90, 1991. (Cité en page 29.)

[GP90] G.M. Guisewite and P.M. Pardalos. Minimum concave-cost network flow problems : Applications, complexity, and algorithms. *Annals of Operations Research*, 25(1) :75–99, 1990. (Cité en pages 79 et 80.)

[GS03] M. Gupta and S. Singh. Greening of the internet. In *SIGCOMM*, 2003. (Cité en page 78.)

[HMS89] M. C. Heydemann, J.c. Meyer, and D. Sotteau. On forwarding indices of networks. *Discrete Appl. Math.*, 23 :103–123, May 1989. (Cité en page 82.)

[IOR+10] F. Idzikowski, S. Orlowski, C. Raack, H. Woesner, and A. Wolisz. Saving energy in IP-over-WDM networks by switching off line cards in low-demand scenarios. In *ONDM*, 2010. (Cité en page 78.)

[JM03] B. G. Józsa and M. Makai. On the solution of reroute sequence planning problem in MPLS networks. *Computer Networks*, 42(2) :199 – 210, 2003. (Cité en page 13.)

[JMRBB09] A. Jean-Marie, X. Roche, A-E. Baert, and V. Boudet. Combinatorial designs and availability. Technical Report RR 7119, INRIA, 2009. (Cité en pages 5, 134 et 135.)

[JMT07] B. Jaumard, C. Meyer, and B. Thiongane. Comparison of ilp formulations for the rwa problem. *Optical Switching and Networking*, 4(3-4) :157 – 172, 2007. (Cité en page 12.)

[JS03] N. Jose and A.K. Somani. Connection rerouting/network reconfiguration. In *Design of Reliable Communication Networks*. IEEE, 2003. (Cité en pages 16, 29 et 76.)

[Kan92] V. Kann. *On the Approximability of NP-complete Optimization Problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm., 1992. see also : http ://www.csc.kth.se/ viggo/wwwcompendium/node19.html. (Cité en pages 21 et 74.)

[Kin92] N. G. Kinnersley. The vertex separation number of a graph equals its pathwidth. *Information Processing Letters*, 42(6) :345–350, 1992. (Cité en page 25.)

[Klo08] O. Klopfenstein. Rerouting tunnels for MPLS network resource optimization. *European Journal of Operational Research*, 188(1) :293 – 312, 2008. (Cité en page 13.)

[KMP08] Ralf Klasing, Nelson Morales, and Stéphane Pérennes. On the complexity of bandwidth allocation in radio networks. *Theoretical Computer Science*, 406(3) :225 – 239, 2008. (Cité en page 111.)

[KP86] M. Kirousis and C.H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47(2) :205–218, 1986. (Cité en pages 24 et 26.)

[LaP93] A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the Association for Computing Machinery*, 40(2) :224–245, 1993. (Cité en pages 24, 25 et 27.)

[LCZ05] Qiao Lian, Wei Chen, and Zheng Zhang. On the impact of replica placement to the reliability of distributed brick storage systems. In *International Conference on Distributed Computing Systems (ICSCS'05)*, pages 187–196, Los Alamitos, CA, USA, 2005. IEEE Computer Society. (Cité en page 133.)

[LHA94] J.-F.P. Labourdette, G.W. Hart, and A.S. Acampora. Branch-exchange sequences for reconfiguration of lightwave networks. *IEEE Transactions on Communications*, 42(10) :2822–2832, Oct 1994. (Cité en page 13.)

[LJ99] Hen-Ming Lin and Jing-Yang Jou. Computing minimum feedback vertex sets by contraction operations and its applications on cad. In *Computer Design, 1999. (ICCD '99) International Conference on*, pages 364 –369, 1999. (Cité en page 21.)

[LL96] K.-C. Lee and V.O.K. Li. A wavelength rerouting algorithm in wide-area all-optical networks. *IEEE/OSA Journal of Lightwave Technology*, 14(6) :1218–1229, June 1996. (Cité en page 13.)

[LP86] L. Lovász and M.D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland, 1986. (Cité en pages 111 et 113.)

[LZ08] F. Li and B. Zhou. Minimal energy of unicyclic graphs of a given diameter. *Journal of Mathematical Chemistry*, 43(2) :476–484, February 2008. (Cité en page 49.)

[MCK+09] K. Manousakis, K. Christodoulopoulos, E. Kamitsas, I. Tomkos, and E. Varvarigos. Offline impairment-aware routing and wavelength assignment algorithms in translucent WDM optical networks. *IEEE/OSA Journal of Lightwave Technology*, 27(12) :1856–1877, 2009. (Cité en page 73.)

[MG92] J Misra and David Gries. A constructive proof of vizing's theorem. *Information Processing Letters*, 41, 1992. (Cité en page 124.)

[MHG+88] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the Association for Computing Machinery*, 35(1) :18–44, 1988. (Cité en page 23.)

[MM99] G. Mohan and C.S.R. Murthy. A time optimal wavelength rerouting algorithm for dynamic trafficin WDM networks. *IEEE/OSA Journal of Lightwave Technology*, 17(3) :406–417, March 1999. (Cité en page 13.)

[MSBR08] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. Energy Aware Network Operations. In *HP Labs*, 2008. (Cité en page 78.)

[MSBR09] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A Power Benchmarking Framework for Network Devices. In *IFIP NETWORKING 2009*, pages 795–808, may 2009. (Cité en pages 4 et 78.)

[MSZ06] Eytan Modiano, Devavrat Shah, and Gil Zussman. Maximizing throughput in wireless networks via gossiping. *SIGMETRICS Perform. Eval. Rev.*, 34(1) :27–38, 2006. (Cité en page 112.)

[MT09] R. Mihai and I. Todinca. Pathwidth is NP-hard for weighted trees. In X. Deng, J. E. Hopcroft, and J. Xue, editors, *3rd International Workshop on Frontiers in Algorithmics (FAW)*, volume 5598 of *Lecture Notes in Computer Science*, pages 181–195, Hefei, China, June 2009. Springer. (Cité en page 49.)

[NC05] Bruce Nordman and Ken Christensen. Reducing the energy consumption of networked devices. www.csee.usf.edu/~christen/energy/lbnl_talk.pdf, 2005. (Cité en page 78.)

[NIRW08] L. Nedevschi, S.and Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing Network Energy Consumption via Rate-Adaptation and Sleeping. In *NSDI*, 2008. (Cité en page 78.)

[OFGZ08] Yury Orlovich, Gerd Finke, Valery Gordon, and Igor Zverovich. Approximability results for the maximum and minimum maximal induced matching problems. *Discrete Optimization*, 5(3) :584 – 593, 2008. (Cité en page 112.)

[Par78] T. D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, volume 642 of *Lecture Notes in Mathematics*, pages 426–441. Springer, Berlin, 1978. (Cité en pages 24 et 31.)

[Pel90] David Peleg. Time-optimal leader election in general networks. *J. Parallel Distrib. Comput.*, 8(1) :96–99, 1990. (Cité en page 38.)

[PHH+00] S.-L. Peng, C.-W. Hob, T.-S. Hsu, M.-T. Ko, and C. Y. Tanga. Edge and node searching problems on trees. *Theoretical Computer Science*, 240(2) :429–446, June 2000. (Cité en pages 25, 26 et 31.)

[PVC+09] B. Puype, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester. Power Reduction Techniques in Multilayer Traffic Engineering. In *ICTON*, 2009. (Cité en page 78.)

[RS83] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1) :39–61, 1983. (Cité en pages 24 et 25.)

[RSS09] Shreevatsa Rajagopalan, Devavrat Shah, and Jinwoo Shin. Network adiabatic theorem : an efficient randomized protocol for contention resolution. In *SIGMETRICS '09 : Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 133–144, New York, NY, USA, 2009. ACM. (Cité en page 112.)

[Sch90] P. Scheffler. A linear algorithm for the pathwidth of trees. In R. Henn R. Bodendiek, editor, *Topics in Combinatorics and Graph Theory*, pages 613–620. Physica-Verlag Heidelberg, 1990. (Cité en page 29.)

[Sko03] K. Skodinis. Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. *Journal of Algorithms*, 47(1) :40–59, 2003. (Cité en page 29.)

[SL05] M. Saad and Zhi-Quan Luo. Reconfiguration with no service disruption in multifiber WDM networks. *IEEE/OSA Journal of Lightwave Technology*, 23(10) :3092–3104, October 2005. (Cité en page 13.)

[Sol09] F. Solano. Analyzing two different objectives of the WDM network reconfiguration problem. In *IEEE Global Communications Conference (Globecom)*, 2009. (Cité en page 22.)

[SPa10] F. Solano and M. PioÌ andro. Lightpath reconfiguration in wdm networks. *Optical Communications and Networking, IEEE/OSA Journal of*, 2(12) :1010 –1021, december 2010. (Cité en pages 29 et 76.)

[Spi64] Kurt Spielberg. On the fixed charge transportation problem. In *Proceedings of the 1964 19th ACM national conference*, pages 11.101–11.1013, New York, NY, USA, 1964. ACM. (Cité en page 80.)

[SV82] Larry J. Stockmeyer and Vijay V. Vazirani. Np-completeness of some generalizations of the maximum matching problem. *Inf. Process. Lett.*, 15(1) :14–19, 1982. (Cité en page 111.)

[Tas97] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. *IEEE Transactions on Information Theory*, 43(3) :1067–1073, 1997. (Cité en page 111.)

[TE90] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Conference on Decision and Control*, pages 2130–2132 vol.4, 1990. (Cité en pages 111 et 125.)

[Wan09] P.-J Wan. Multiflows in multihop wireless networks. In *MobiHoc '09*, pages 85–94. ACM, 2009. (Cité en page 108.)

[Wei] Eric W. Weisstein. Wiener index. From MathWorld–A Wolfram Web Resource, http ://mathworld.wolfram.com/WienerIndex.html. (Cité en pages 94 et 98.)

[Wie47] H. Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1) :17–20, 1947. (Cité en pages 94 et 98.)

[YJ71] B. Yaged Jr. Minimum cost routing for static network models. *Networks*, 1(2) :139–172, 1971. (Cité en page 79.)

[ZRG06] Zhongzhi Zhang, Lili Rong, and Chonghui Guo. A deterministic small-world network created by edge iterations. *Physica A : Statistical Mechanics and its Applications*, 363(2) :567–572, May 2006. (Cité en page 6.)

[ZYWS07] J. Y. Zhang, O. W. W. Yang, J. Wu, and M. Savoie. Optimization of semi-dynamic lightpath rearrangements in a WDM network. *IEEE Journal on Selected Areas in Communications*, 25(9) :3–17, December 2007. (Cité en page 13.)