UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DE L'ALLOCATION DE RESSOURCES DANS UN RÉSEAU DE TÉLÉCOMMUNICATIONS PAR COLORATION IMPROPRE DE GRAPHES

ROMAIN MONTAGNÉ
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AVRIL 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMISATION DE L'ALLOCATION DE RESSOURCES DANS UN RÉSEAU DE TÉLÉCOMMUNICATIONS PAR COLORATION IMPROPRE DE GRAPHES

présentée par : MONTAGNÉ Romain
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. GENDREAU Michel, Ph. D., président
M. HERTZ Alain, Doctorat ès Sc., membre et directeur de recherche
M. GAGNON François, Ph. D., membre et codirecteur de recherche
M. CARDINAL Christian, Ph. D., membre
Mme LINHARES SALES Cláudia, Doctorat, membre externe

REMERCIEMENTS

J'aimerais remercier tous ceux qui ont contribué à l'élaboration de cette thèse, de près ou de loin.

Je pense tout d'abord à mes directeurs de thèse, François et Alain. Merci de m'avoir guidé et encadré comme vous l'avez fait. François, j'ai particulièrement apprécié ton ouverture d'esprit et ton aide précieuse pour tout l'aspect télécommunications. Sans aucun doute, ta capacité à faire le lien entre le monde des graphes et celui des télécommunications a été décisive. Alain, travailler à tes côtés a été pour moi une grande source d'inspiration, j'ai beaucoup apprécié la façon dont nous avons pu, ensemble, développer certaines idées de cette thèse. Ta touche pédagogique est unique, je vais essayer à l'avenir d'y être fidèle. Merci à tous les deux pour la liberté que vous m'avez laissée durant ces années de travail.

Une pensée particulière pour tous les professeurs de ma scolarité qui m'ont marqué et donné le goût d'apprendre, de chercher, de persévérer : Daniel Carron, Thierry Carette, Claude Lemaire, Marie-Jo Gervasi, Jean Guérin, Dominique Orban, Michel Gamache, Alain Hertz.

Je remercie également les professeurs qui m'ont permis de faire mes premiers pas dans le monde de l'enseignement durant cette thèse. Jean Guérin, Michel Gamache, Dominique Orban, merci pour votre confiance.

Je souhaite également remercier tous mes camarades et amis du GERAD ou de Polytechnique qui m'ont accompagné, parfois aidé à franchir certains obstacles auxquels on fait inéluctablement face lors d'un doctorat : Georges, Samuel, Thibault, Mouad, Nadir, Atoosa, Lê, Simon, Jojo, JB, Sara.

Enfin, je remercie chaleureusement ma famille et mes proches, ils se reconnaîtront. Merci pour votre soutien, pour votre confiance, vous êtes aussi une source d'inspiration pour moi.

RÉSUMÉ

Dans cette thèse, nous traitons le problème d'allocation de ressources pour les réseaux de télécommunications, dont le développement ne cesse de s'intensifier tant la demande en transmission de données est élevée à travers le monde. Malheureusement, ces ressources sont limitées et il devient de plus en plus difficile de satisfaire les besoins des utilisateurs, d'où l'intérêt de mettre en œuvre des méthodes mathématiques adaptées à cette problématique. De telles techniques ont été développées depuis l'apparition des premiers réseaux dans les années 1970, mais l'évolution de la technologie sans fil et l'amélioration des puissances de calculs informatiques nécessitent de constamment mettre à jour les modèles utilisés, ainsi que les méthodes pour les résoudre ; cette thèse s'inscrit dans cette dynamique.

Nous proposons ici un nouveau modèle d'assignation de canaux qui suppose l'existence d'une entité capable de centraliser les données recueillies au niveau des usagers, à l'échelle métropolitaine. En particulier, nous proposons de maximiser le nombre d'usagers pouvant être connectés simultanément au réseau. Nous verrons en quoi ce modèle est nouveau mais tout à fait réaliste compte tenu des modèles existants.

L'un des aspects originaux de cette thèse est le fait de remettre la théorie des graphes au goût du jour pour traiter ce problème d'affectation. Les notions de stabilité et de coloration impropre sont en effet intimement liées à ce problème. Nous verrons également qu'une généralisation de la fonction de Lovász permet de dériver une borne supérieure sur l'objectif, plus serrée que la relaxation linéaire du modèle en nombres entiers associé.

La théorie des graphes est un outil de modélisation très puissant, intimement liée à l'algorithmique inhérente aux problèmes combinatoires complexes. Nous verrons comment prendre à contre-pied l'explosion combinatoire de ce problème $\mathcal{NP}$-difficile sans compromettre la qualité des solutions obtenues, en généralisant les notions de degré et de degré de saturation d'un sommet, à la base des heuristiques de coloration les plus populaires. Les algorithmes développés seront analysés théoriquement via les notions de complexité, de garantie et de plus petit graphe sous-optimal, mais aussi numériquement : ceux-ci seront comparés aux méthodes d'affectation classiques, à savoir la réutilisation de fréquence fractionnée. Enfin, nous étendrons le modèle proposé sur un horizon de temps, et traiterons le problème lorsque les données ne sont pas entièrement connues à l'avance mais apparaissent au fur et à mesure.

En résumé, réseaux de télécommunications, théorie des graphes, algorithmique et explosion combinatoire sont les ingrédients principaux de cette thèse : il s'agit bel et bien de recherche opérationnelle !

ABSTRACT

Since the 1970's wireless networks have constantly been striving to meet consumers increasing demands, while the necessary resources (electromagnetic frequencies) are becoming increasingly scarce. In this thesis, we address the problem of allocating a channel to a maximum number of mobiles, in order to maximize the number of users that are simultaneously connected to the network. This is a typical assignment problem, for which operations research provides the adequate tools.

We propose a new model based on the existence of a calculator capable of dealing with data at a metropolitan scale. Although ambitious, this scheme remains realistic given the latest developments of cloud computing and virtualization technologies. Most importantly, part of the originality of this model is its link with graph theory. We will see how the concepts of stability and improprer weighted coloring are intimately related to the channel assignment problem. Also, we will show how a generalization of Lovász's function can be used to derive an upper bound on the objective function, tighter than the linear relaxation of the associated integer model.

As one knows, a graph theory approach is often related to combinatorics and its difficulties, namely the combinatorial explosion of any $\mathcal{NP}$-hard problem. We will see how to cope with this aspect, at the crossroads between fundamental and applied research. In particular, we will generalize the concepts of degree and saturation degree of a vertex, which are essential components of most of the popular vertex coloring heuristics. The proposed algorithms are analyzed both theoretically - through computational complexity, performance guarantee and smallest suboptimal graphs -, and numerically: a set of test results are reported in which the heuristics are compared to classical allocation techniques based on fractional frequency reuse. Last but not least, the model is extended over a rolling horizon and on-line algorithms are proposed to tackle the problem when data is not known in advance but appears iteratively.

In summary, wireless networks, graph theory, algorithms and combinatorial explosion are the main ingredients to this thesis which definitely lies in the scope of operations research.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AMPS	Advanced Mobile Phone System
ARPD	Average Relative percentage Deviation
PB	Probabilité de Blocage
CC	Coupes de Couverture
CG	Génération de Colonnes
CLC	Coupes de CLiques
CPU	Central Processing Unit
CS	Connected Sequential
DECT	Digital European Cordless Telecommunications
DIMACS	Center for Discrete Mathematics and Theoretical Computer Science
DSAT	Dsatur
FFR	Fractional Frequency Reuse
GIS	Greedy Independent Set
IC	Contraintes d'Incompatibilité
INRIA	Institut National de Recherche en Informatique et en Automatique
LTE	Long Term Evolution
LUB	Borne supérieure de Lovász
OC	Contraintes d'Ordre
PMR	Problème Maître Restreint
RAM	Random Acces Memory
RLF	Recursive Largest First
RS	Random Sequential
SINR	Signal to Interference plus Noise Ratio
SL	Smallest Last
SLF	Saturation Largest First
SUB	Sous-Problème
TAPD	Total Absolute Percentage Deviation
UTI	Union Internationale des Télécommunications
WC	Coupes de poids
WLAN	Wireless Local Area Network
WP	Welsh-Powell

CHAPITRE 1 INTRODUCTION

La téléphonie mobile est en train de conquérir la Terre. Les études tirées de [1, 2] annonçaient en 2014 que le nombre d'abonnements de téléphonie mobile s'élevaient à 6,8 milliards pour 7,2 milliards d'habitants sur notre planète. Le rapport d'Ericsson [1] estime que d'ici 5 ans la consommation de données échangées atteindra 2,5 Go en moyenne par mois et par utilisateur, et qu'à cette date le très haut débit mobile couvrira 65 % de la population mondiale. Par ailleurs, les ressources du réseau de téléphonie mobile (les canaux de communication, ou fréquences d'ondes électromagnétiques) sont en nombre limité : seule une partie du spectre électromagnétique est dédiée à la téléphonie mobile, le reste étant reservé à d'autres applications (service maritime, applications militaires, aéronautique, etc.) toutes aussi gourmandes en radio fréquences. En réponse à cette demande croissante des consommateurs à l'égard d'une couverture étendue et de débits supérieurs, le réseau de téléphonie mobile ne cesse d'évoluer depuis son apparition dans les années 1970, notamment avec les techniques de multiplexage spatial qui permettent à plusieurs utilisateurs de partager les ressources du réseau. On dispose aujourd'hui d'un réseau de quatrième génération (4G) dans lequel un canal peut être emprunté par plusieurs usagers, du moment que les interférences (causées par la présence simultanée de ces usagers sur le canal) ne dépassent pas un certain seuil [3]. La façon dont sont assignés les canaux de communication aux utilisateurs influe directement sur la présence ou non d'interférences, et sur le nombre maximum d'usagers pouvant communiquer simultanément sur le réseau. Il devient donc primordial de disposer de méthodes efficaces pour pouvoir effectuer de telles affectations.

1.1 Problématique

Dans un tel contexte, la problématique est de savoir comment il faut assigner les canaux de communication aux utilisateurs, de façon à ce qu'un maximum d'entre eux puissent être connectés au réseau. Étant donné que les ressources sont en nombre restreint, on ne peut évidemment pas attribuer un canal à chaque utilisateur. Inversement, si trop d'utilisateurs empruntent le même canal, cela crée des interférences qui dégradent la qualité des signaux échangés. Il s'agit donc d'un problème combinatoire complexe qui consiste à satisfaire le mieux possible une demande sujette à des contraintes de ressource. La recherche opérationnelle est justement la branche des mathématiques qui, développée essentiellement au cours du siècle dernier, fournit les outils adéquats pour traiter ce type de problématique.

1.2 Cadre de l'étude : la recherche opérationnelle

La recherche opérationnelle peut se définir comme *la mise en oeuvre de méthodes scientifiques, essentiellement mathématiques, en vue de prendre la meilleure décision possible*[1]. Cet outil d'aide à la décision passe nécessairement par l'élaboration d'un modèle conceptuel où sont clairement identifiées les variables (ou marges de manœuvre du décideur), les contraintes liées à la nature du problème (le cadre qui délimite ces marges de manœuvre), ainsi qu'un ou plusieurs objectifs (ce que vise le décideur). Ce modèle est alors résolu algorithmiquement ; l'interprétation de la solution obtenue correspond à la stratégie à mettre en œuvre pour résoudre le problème initial. Ce processus, résumé ci-dessous à la Figure 1.1, fixe le cadre de ces travaux qui comprennent essentiellement des éléments de modélisation du problème d'affectation dont il s'agit ici, ainsi que des méthodes algorithmiques pour résoudre ce modèle.

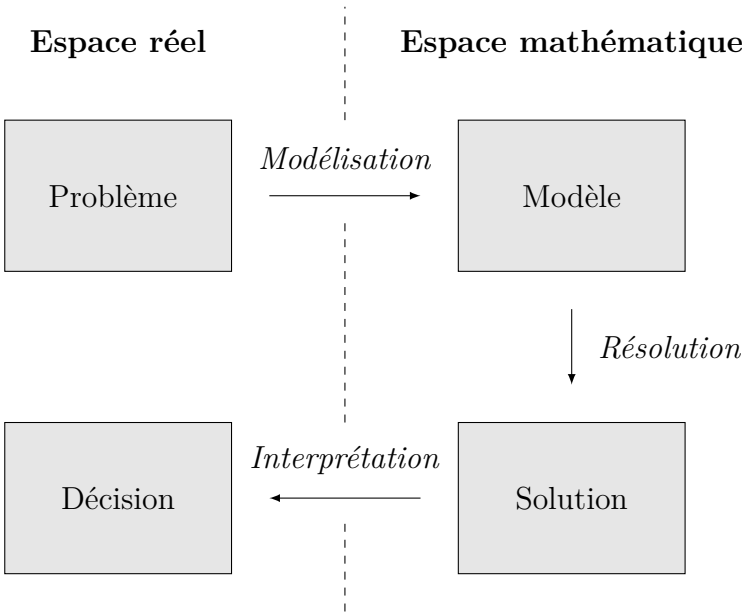


Figure 1.1 Recherche opérationnelle : démarche et étapes

1.3 Structure générale de la thèse et contributions

Le Chapitre 2 est dédié à la description du problème d'allocation de ressources que l'on se propose de traiter, via une revue de littérature critique. Nous introduirons les notions de télécommunications nécessaires à la compréhension du problème et ferons une synthèse des

1. Source : `http://www.roadef.org`

méthodes existantes pour le résoudre, ce qui permettra de mettre en lumière l'originalité et l'unicité du modèle proposé. Celui-ci s'appuie sur la théorie des graphes, en particulier sur la coloration impropre des sommets sur un graphe complet, orienté, pondéré ; ces concepts seront définis de façon formelle. Enfin nous verrons en quoi le problème est intrinsèquement difficile au sens de la théorie de la complexité : à moins que $\mathcal{P} = \mathcal{NP}$, aucun algorithme ne peut prétendre le résoudre de façon exacte sans que le temps de résolution n'augmente de façon exponentielle avec la taille du problème.

Pour s'attaquer à la résolution de tels problèmes, deux options se présentent : soit on utilise des méthodes exactes qui garantissent de trouver une solution optimale, en sachant que l'explosion combinatoire est inévitable et que l'on ne pourra pas résoudre des instances trop grandes, soit on renonce à cette garantie et l'on utilise des méthodes heuristiques qui, à défaut de pouvoir trouver systématiquement la solution optimale, en trouvent rapidement de bonne qualité avec une probabilité élevée.

Le Chapitre 3 traite la première approche. Plus précisément, trois différentes formulations en nombres entiers sont proposées, dont une basée sur la décomposition de Dantzig-Wolfe. Celles-ci sont comparées dans un premier temps théoriquement à travers des relations d'inclusion entre les polyèdres engendrés par les contraintes, puis numériquement par branch-and-bound, branch-and-cut et branch-and-price sur des instances aléatoires. On montre ensuite comment améliorer ces formulations au moyen de coupes et d'une borne supérieure basée sur la fonction généralisée de Lovász. Une série de tests numériques mettent en évidence dans quelle mesure ces éléments permettent de diminuer le nombre de nœuds de branchements et par conséquent le temps de résolution : sur certaines instances, ces derniers sont divisés par 10. Ces propositions ainsi que leur analyse sont nouvelles et constituent donc le premier ensemble de contributions de cette thèse.

La seconde approche est développée dans le Chapitre 4. Les notions de degré et de degré de saturation d'un sommet sont généralisées, et les heuristiques populaires s'appuyant sur ces concepts (Largest First, Saturation Largest First, Recursive Largest First) sont adaptées au problème de coloration dont il s'agit ici. La performance de ces algorithmes est évaluée théoriquement à partir de leurs complexités, de leurs plus petits graphes sous-optimaux, et numériquement par comparaison avec les méthodes exactes, et avec les méthodes utilisées par les fournisseurs de services mobiles, à savoir la *réutilisation de fréquence fractionnée* (*fractional frequency reuse*, en anglais). Nous verrons également comment exploiter la structure des réseaux de télécommunications permet de gagner en précision. Il s'avère que le modèle de graphe proposé couplé à des heuristiques en $\mathcal{O}(n^2)$ permettent des gains allant jusqu'à 50%.

Enfin, le Chapitre 5 est une étude du modèle étendu sur un horizon de temps, où les données

du problème ne sont pas entièrement connues à l'avance et apparaissent étape par étape. Le problème est ainsi traité de façon dynamique, les décisions au cours du temps étant irrévocables. Les meilleures garanties de performance théoriques possibles sont étudiées, et différentes politiques de priorité couplées aux heuristiques précédentes sont comparées numériquement. Nous étudions également une relaxation du problème où il est permis de revenir sur certaines décisions moyennant une pénalité, donnant lieu à un problème bi-objectif. Deux approches sont proposées pour traiter cette relaxation : une à l'aide d'algorithmes dits de réparation, une autre par recherche locale. L'efficacité de ces algorithmes est évaluée sur les instances de coloration DIMACS, et sur des graphes aléatoires orientés et pondérés.

Nous verrons en guise de conclusion les limites du modèle proposé et des algorithmes pour le résoudre, ce qui ouvre la voie à de nouveaux axes de recherche possibles.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre est structuré en deux parties principales. Dans un premier temps, nous passons en revue les différentes méthodes d'assignation de canaux depuis l'apparition des premiers réseaux de télécommunications et montrons en quoi la méthode proposée ici est originale. Une seconde partie met en lumière les éléments nécessaires pour exprimer et résoudre ce problème à travers la théorie des graphes.

2.1 Réseaux de télécommunications : définitions et notions de base

Commençons par introduire certaines notions fondamentales portant sur les réseaux de télécommunications. Il s'agit dans un premier temps de saisir le contexte dans lequel s'inscrivent ces travaux, ainsi que les notions nécessaires pour traiter le problème d'affectation de ressources. Pour comprendre ces notions plus en profondeur, le lecteur peut se référer par exemple à [4].

Canaux de communication

Un canal de communication est un médium de transmission d'information entre un émetteur et un récepteur. Tel qu'évoqué en introduction, en télécommunications ce médium est constitué d'une onde électromagnétique (typiquement sinusoïdale) qui par modulation transporte le signal entre l'émetteur et le récepteur. La Figure 4.5 ci-dessous illustre cette notion : le signal à transmettre (représenté en bleu) est « porté » par une onde électromagnétique de plus haute fréquence (en rouge).

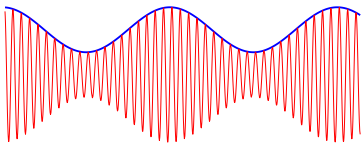


Figure 2.1 Une porteuse de signal transporte un message

Cette porteuse est caractérisée par sa fréquence qui est reliée par une expression affine à un numéro de canal ; on parlera donc ici indifféremment du canal, de sa fréquence ou de porteuse. Les techniques de multiplexage apparues vers 1980 ont permis à plusieurs utilisateurs d'emprunter un même canal simultanément. Le multiplexage temporel (Time Division Multiple

Access) consiste à entrelacer dans le temps des échantillons de signaux discrétisés. Le pas de temps, fixé notamment à l'aide du théorème d'échantillonnage de Nyquist-Shannon, est tel que l'oreille ne perçoit pas que le signal est découpé. Cette technique permet de faire circuler jusqu'à une trentaine de signaux sur un même canal. Le multiplexage fréquentiel (Frequency Division Multiple Access) consiste lui à découper les canaux en plusieurs sous-canaux plus fins. Généralement, ces deux types de multiplexages sont utilisés de façon conjointe pour maximiser le nombre de données numériques pouvant être transportées.

Finitude du nombre de canaux

Le nombre de canaux disponbiles pour la téléphonie mobile est limité. En effet, le spectre des radiofréquences est une ressource partagée entre autres par les services de communication et de transport, la défense, les prévisions météorologiques, l'agriculture. Au Canada, le ministère du gouvernement chargé de la politique industrielle (Industrie Canada) et l'Union Internationale des Télécommunications (UIT) gèrent ensemble l'attribution de ces ressources. Par exemple, le service sans fil commercial dispose (au Canada) des bandes 824-849 MHz et 869-894 MHz [1].

Stations de base

Chaque opérateur dispose de son propre parc de stations dites de transmission (ou de base) réparties sur tout le territoire, chargées de la liaison radio entre les usagers et le réseau. La Figure 5.1 illustre la répartition de celles de l'opérateur Bell sur l'île de Montréal .

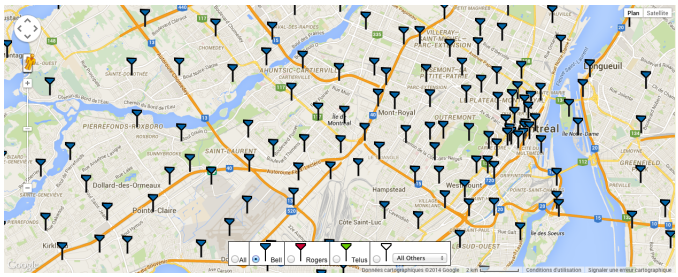


Figure 2.2 Répartition des stations de base de l'opérateur Bell sur l'île de Montréal [2]

Un usager est connecté au réseau s'il dispose d'une porteuse pouvant transporter son message entre son téléphone mobile et la station de base la plus proche. De la même façon, le récepteur

1. L'ensemble des attributions du spectre est accessible à la page web suivante : `www.ic.gc.ca/eic/site/smt-gst.nsf/fra/h_sf01678.html`.
2. Source :`www.ertyu.org/steven_nikkel/cancellsites.html`

peut recevoir l'information si un canal pouvant transmettre l'information entre une station de base et son téléphone mobile lui a bien été attribué. Ces stations de base, dont le rayon de portée peut atteindre 30 Km, sont pilotées par un contrôleur de stations chargé notamment de gérer la répartition des ressources entre ces stations. Enfin, le centre de communication du service mobile, géré directement par l'opérateur téléphonique, relie le réseau à Internet et au réseau de téléphone fixe.

Puissance des signaux

La puissance P_{ip} d'un signal émis par un usager i et reçue à une station de base p peut s'écrire sous la forme suivante [5] :

$$P_{ip} = \frac{\alpha\, r_{ip}}{{d_{ip}}^{\gamma}}$$

où :
- α est une constante qui prend en compte la configuration et la position des antennes ;
- r_{ip} est une variable aléatoire utilisée pour traduire l'affaiblissement de propagation du signal ;
- d_{ip} est la distance Euclidienne entre l'usager i et sa station de base p ;
- γ est le facteur d'atténuation qui varie de 2 (dans le vide ou en espace libre) à 4 (dans les zones urbaines).

Rapport signal-bruit

Les interférences pouvant perturber la réception des signaux peuvent être classées en trois catégories :
- le bruit ambiant ;
- les interférences inter-canaux : ce sont les interférences se produisant entre les signaux émettant sur des canaux différents, mais proches géographiquement. Celles-ci sont négligeables si les fréquences utilisées entre stations de bases voisines sont suffisamment éloignées les unes des autres ;
- les interférences de canal : il s'agit des interférences se produisant entre les signaux émettant sur un même canal. Ce sont les perturbations les plus contraignantes à partir desquelles la qualité d'un signal est évaluée. En effet, celle-ci est mesurée par le rapport signal-bruit (SINR), i.e. le rapport entre la puissance du signal émis P_{ip} et la puissance des interférences de canal (à laquelle on ajoute le bruit $\mathcal{P}_0$). En notant $c(i)$ le canal affecté à l'usager i, ce rapport s'écrit de la façon suivante :

$$\mathrm{SINR}_{ip} = \frac{P_{ip}}{\sum_{j \neq i | c(j)=c(i)} P_{jp} + \mathcal{P}_0}.$$

Les réseaux actuels tolèrent un rapport signal-bruit supérieur à 3 dB.

Moyennant quelques hypothèses simplificatrices [4], le SINR d'un usager i connecté à une station de base p de rayon R peut être estimé en fonction du facteur d'atténuation γ et de la distance d_{pq} entre sa station p et la station q la plus proche émettant sur les mêmes canaux que p. Plus précisément, si l'on ne considère que les interférences de canal de premier niveau (provenant des stations de base les plus proches), et que les positions des stations de base définissent un maillage hexagonal dans le plan, alors on a l'approximation suivante :

$$\mathrm{SINR}_{ip} \approx \frac{1}{6} \left(\frac{d_{pq}}{R} \right)^{\gamma}. \tag{2.1}$$

Si l'on note $\frac{1}{\theta}$ le SINR minimum en deçà duquel la communication n'est plus garantie, on peut alors déduire que deux stations qui utilisent simultanément un même canal doivent être espacées d'une distance au moins égale à $R(\frac{6}{\theta})^{1/\gamma}$. Nous verrons ci-après que de nombreuses méthodes d'allocation de canaux se basent sur ce principe.

Assignation de canaux

Il existe deux méthodes principales permettant de réaliser les affectations canal-utilisateur de façon à maintenir un SINR suffisamment élevé : les affectations dites *statiques* et les affectations *dynamiques*. Les affectations statiques sont basées sur une partition de l'espace en cellules (chaque station de base est le centre d'une cellule), qui sont chacune affectées d'un certain nombre de canaux. Typiquement, deux cellules adjacentes vont avoir accès à des canaux différents, ce qui garantit que deux appels utilisant le même canal soient séparés par au moins une cellule, ce qui est généralement suffisant pour garantir un SINR supérieur au seuil requis. Au sein de la cellule, les usagers peuvent utiliser n'importe quel canal disponible, dans les limites imposées par les techniques de multiplexage. La plupart des systèmes cellulaires adoptent ce type d'affectations [6]. Ces schémas présentent l'avantage d'être simples et robustes, mais ne permettent pas de s'adapter aux variations du trafic, contrairement aux affectations dynamiques. Ces dernières ne prédéterminent pas une attribution de canaux aux stations, mais gèrent un appel en affectant à l'usager n'importe quel canal parmi tous ceux lui garantissant un SINR acceptable. Au prix d'une plus grande complexité, ces schémas sont plus flexibles. Cependant, cette stratégie s'avère moins efficace lorsque le trafic est très dense. La section qui suit est une synthèse de ces différentes techniques.

2.2 Assignation de canaux

2.2.1 Assignation statique

Les premiers problèmes d'affectation statique de canaux remontent aux années 1960. Ils consistent à assigner un groupe de fréquences aux stations de base, qui peuvent alors desservir les usagers se trouvant suffisamment proches de la station. Afin de limiter les interférences, ces assignations doivent respecter certaines contraintes : une différence minimum (typiquement 3) entre les fréquences attribuées à une même station, une différence minimum (typiquement 2) entre les fréquences attribuées à deux stations voisines. De plus, les canaux assignés à une station doivent appartenir à un ensemble prédéterminé, qui peut être distinct pour chaque station, en fonction du trafic estimé autour de celle-ci.

Répétition de patrons génériques de cellules

La façon dont la majorité des fournisseurs de services mobiles [6] traitent ce problème consiste à résoudre le problème sur un réseau réduit de quelques cellules appelé motif, et de l'utiliser comme patron générique pour couvrir tout le réseau initial [4]. Typiquement, les cellules utilisées sont des hexagones, car ce sont les figures qui se rapprochent le plus du cercle (qui correspond à la zone de couverture réelle), tout en permettant de couvrir une région plane sans chevauchements, et ceci en minimisant le nombre de figures utilisées (comparativement aux triangles ou aux carrés). La taille de ces patrons est alors déterminée par trois lois. Premièrement, étant donné qu'une cellule hexagonale est adjacente à exactement six autres hexagones répartis de façon uniforme autour de celle-ci, on peut montrer par des considérations géométriques [4] que la taille N des patrons (en nombre de cellules) doit vérifier la relation suivante :

$$N = i^2 + ij + j^2,$$

où i et j sont des entiers naturels. Deux cellules utilisant les mêmes canaux sont alors séparées par i cellules verticalement, et j cellules à $\frac{\pi}{3}$ radians de l'axe vertical (un exemple est illustré à la Figure 2.3). Deuxièmement, en utilisant à nouveau les notations R (le rayon de portée d'une station) et d_{pq} (la distance entre deux stations p et q pouvant émettre sur les mêmes canaux), on a [4] :

$$\frac{d_{pq}}{R} = \sqrt{3N}.$$

Enfin, on a vu que la contrainte garantissant que tout usager ait un SINR supérieur ou égal à un seuil noté $\frac{1}{\theta}$ peut s'écrire comme suit :

$$d_{pq} \geq R \left(\frac{6}{\theta}\right)^{1/\gamma}.$$

En combinant ces trois equations, on obtient la contrainte suivante :

$$(i^2 + ij + j^2) \geq \left\lceil \frac{1}{3} \left(\frac{6}{\theta}\right)^{2/\gamma} \right\rceil.$$

Par exemple, le système cellulaire AMPS (Advanced Mobile Phone System) utilisé initialement aux États-Unis imposait un SINR supérieur ou égal à 18 dB [4], soit $\frac{1}{\theta} = 63$. En prenant le facteur d'atténuation γ égal à 4, on obtient $(i^2 + ij + j^2) \geq \lceil 6.49 \rceil = 7$. Ceci peut être réalisé avec par exemple $i = 1$ et $j = 2$, le patron obtenu est illustré à la Figure 2.3, à gauche. Deux cellules pouvant utiliser les mêmes canaux (appartenant à deux patrons distincts) sont colorées en bleu, à droite.

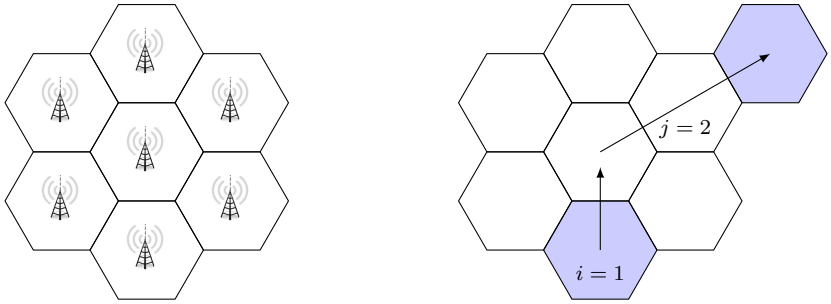


Figure 2.3 Exemple de patron générique

Ces patrons étant déterminés, il faut alors affecter des groupes de canaux à chacune des cellules du patron. En poursuivant l'exemple de l'ancien standard AMPS utilisé aux États-Unis, si l'on suppose qu'un total de 21 canaux sont disponibles, on peut attribuer à chaque cellule C_i $(i = 1, \cdots, 7)$ les canaux i, $i + 7$, $i + 14$.

Bien sûr, ces modèles sont approximatifs dans la mesure où ils supposent que la répartition des stations de base est uniforme et génère un maillage hexagonal du plan. Lorsque les stations sont situées aléatoirement, le maillage obtenu n'est pas régulier (voir la Figure 2.4) et porte le nom de *pavage de Voronoï*. Plus de détails sur ces pavages sont donnés à la Section 2.5.3.

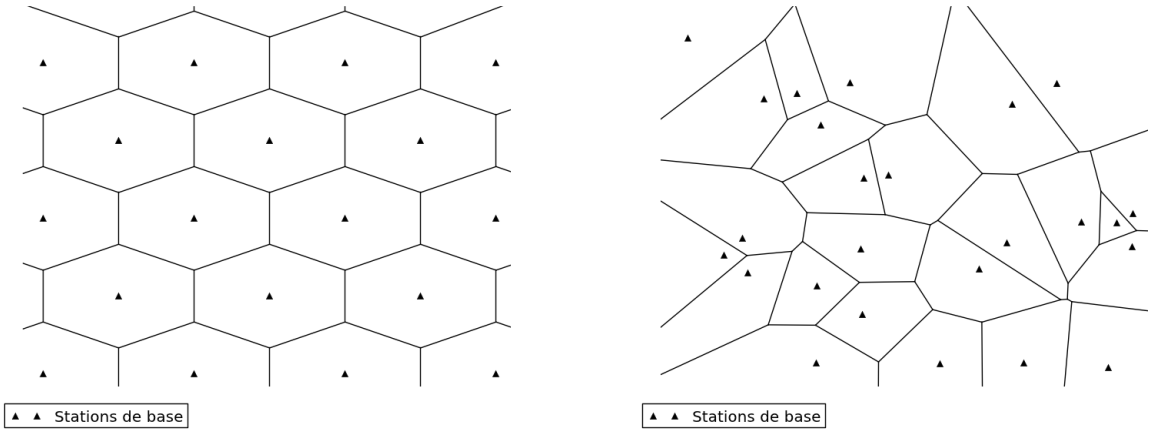


Figure 2.4 Maillage hexagonal et pavage de Voronoï associés aux stations de base

Fonctions objectifs

Ces contraintes étant posées, de nombreuses fonctions objectifs peuvent être considérées pour optimiser l'affectation. Dans les années 1970, le gouvernement facturait le nombre de fréquences à l'unité. Naturellement, les fournisseurs de services mobiles cherchaient alors à minimiser le nombre total de fréquences utilisées [7, 8]. Parfois, c'était plutôt la bande de fréquence utilisée qui était facturée, et l'on cherchait alors à minimiser l'écart entre la plus grande et plus petite fréquence utilisées [9, 10]. D'autres ont cherché à maximiser le nombre de fréquences attribuées aux stations de façon à assurer un taux de service maximal [11, 12], ou encore à minimiser la probabilité qu'un appel soit bloqué aux alentours d'une station [13, 14]. Enfin, une autre fonction objectif possible consiste à minimiser la totalité des interférences sur le réseau afin que la qualité des signaux échangés soit la meilleure possible [15, 16, 17, 18, 19, 20]. Le détail de chacun de ces modèles se trouve dans le rapport [21].

Bien que les modèles cités ici diffèrent par leur fonction objectif, un point commun subsiste entre toutes ces approches : les canaux sont assignés aux stations de base de façon décentralisée. Une fois les canaux assignés aux stations, chaque station va à son tour pouvoir affecter les fréquences aux utilisateurs de sa cellule, sans prendre en compte ce qu'il se passe en dehors de la cellule. Ces modèles présentent un inconvénient majeur : ils ne prennent pas en compte la variation des besoins en canaux due au trafic.

Assignation statique de canaux avec emprunts

L'affectation statique de canaux avec emprunts permet de considérer ces variations. Par exemple, si une cellule est saturée mais que ses voisines sont vides, celle-ci peut leur emprunter des canaux disponibles, à condition que cela ne crée pas d'interférences pour les appels en cours. Le canal emprunté est alors temporairement indisponible dans la cellule prêteuse, ainsi que dans les cellulles voisines de la cellule saturée. Dès que l'appel est terminé, le canal retourne à sa station initiale, et redevient disponible dans les cellules voisines de la cellule saturée. Ces modèles ont été étudiés notamment dans [22, 23, 24, 25, 26] et diffèrent par leur stratégie d'emprunt (la façon de choisir la cellule prêteuse, et le canal à emprunter) ; le rapport [27] détaille et analyse ces différences.

L'aspect dynamique des méthodes avec emprunts permet bel et bien de tenir compte des variations de trafic dans les cellules, mais l'idée centrale reste la même : les canaux sont assignés aux stations de base, et un appel entrant est affecté d'un canal, emprunté ou pas, provenant de la station de base la plus proche. Les fortes variations spatio-temporelles des appels ont amené les opérateurs à étudier de nouvelles méthodes d'affectation, complètement dynamiques, traitant les appels au fur et à mesure sans pré-affectation de canaux aux stations de base. C'est l'objet de la sous-section suivante, qui s'appuie en partie sur le rapport [27].

2.2.2 Assignation dynamique

Les méthodes d'affectation dynamiques de canaux permettent à un usager d'utiliser n'importe quel canal, à condition que celui-ci assure un SINR suffisamment important. Il existe au moins deux grandes familles de méthodes d'affectation dyamiques de canaux : les méthodes centralisées [25, 28, 29, 30], qui assignent un canal parmi l'ensemble de tous les canaux possibles via un contrôleur centralisé, et les méthodes décentralisées [28, 31, 32, 33, 34, 35, 36, 37], où chaque station est une entité indépendante responsable des affectations de sa zone. Celles-ci peuvent se baser sur les données de ses voisines, ou bien directement en s'appuyant sur des mesures locales des signaux des usagers, sans échange d'information entre éléments du réseau. Le système s'autogère alors de façon à maximiser le service du réseau. Le standard DECT (Digital European Cordless Telecommunications) est entièrement conçu suivant cette idée [27]. Contrairement à tous les systèmes précédents, les positions des usagers sont prises en compte, et les choix ne se basent pas sur la géométrie des cellules liée à la distribution des stations de base, qui bien souvent donnent lieu à des distances de séparation très pessimistes, et donc non optimales. Le grand inconvénient de cette méthode est qu'elle ne prend pas en compte les appels en cours. Ainsi, un canal peut être attribué à un usagé et causer une dégradation du SINR des appels en cours, pouvant éventuellement rompre la communication.

Les méthodes dynamiques centralisées nécessitent des infrastructures coûteuses (un contrôleur centralisé notamment) qui ne sont pas encore mises en oeuvre par les fournisseurs de services mobiles actuels. Cependant, avec le développement des environnements virtuels et du Cloud Computing, on peut s'attendre à ce qu'elles soient remises au goût du jour dans les années à venir. L'approche proposée ici s'inscrit dans cette dynamique.

2.3 Nouvelle approche

On peut donc schématiser les différentes méthodes d'affectation de canaux à l'aide de l'arbre de la Figure 2.5. Chaque méthode présente ses avantages et inconvénients. De manière générale, les méthodes statiques sont plus simples, plus robustes aux variations de trafic, mais intrinsèquement non optimales, alors que les méthodes dynamiques sont plus flexibles, au prix d'une plus grande complexité, et moins robustes au delà d'un seuil critique de trafic.

La nouvelle approche proposée ici s'inscrit dans la famille des méthodes dynamiques, en combinant des caractéristiques décentralisées et centralisées. Plus précisément, elle se base sur une centralisation des données obtenues au niveau de l'utilisateur (les SINR). Elle présente donc l'avantage de prendre en compte les positions exactes des utilisateurs (contrairement aux méthodes centralisées classiques), tout en ayant une vue d'ensemble sur le réseau (contrairement aux méthodes décentralisées basées sur des mesures locales du SINR telles que le standard DECT).

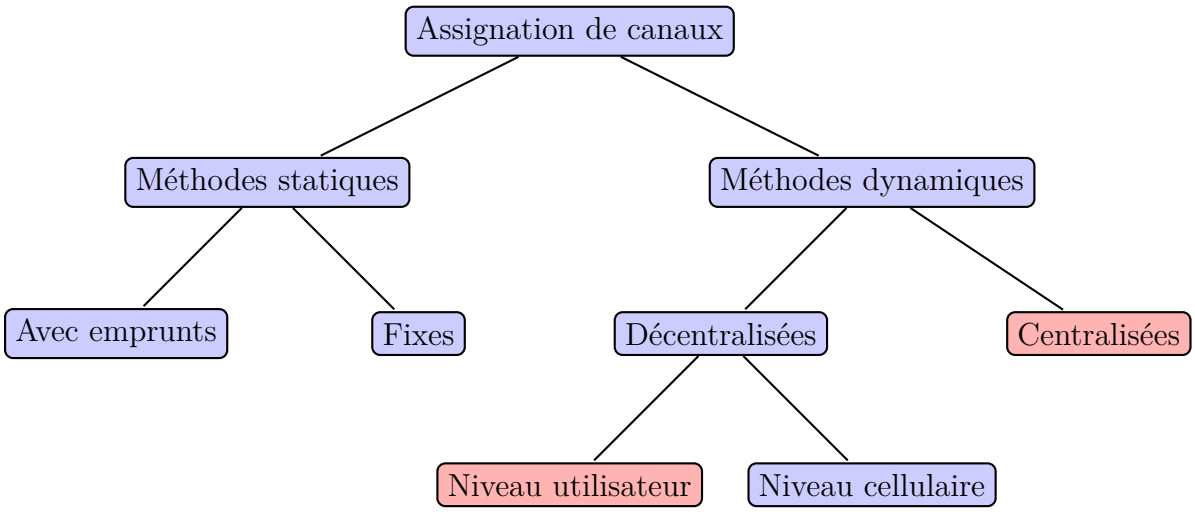


Figure 2.5 Classification des différentes méthodes d'assignation de canaux

Voici son principe : en tant que méthode dynamique, les stations peuvent a priori assigner n'importe quel canal aux utilisateurs se trouvant à proximité. Pour ce faire, le SINR de

chaque utilisateur est mesuré localement, et un canal est candidat si le SINR est supérieur à un certain seuil. Par exemple, le canal $c(i)$ de l'utilisateur i se trouvant à proximité de la station p est candidat si la relation suivante est respectée :

$$\frac{P_{ip}}{\sum_{j \neq i | c(j)=c(i)} P_{jp} + \mathcal{P}_0} \geq \frac{1}{\theta}, \tag{2.2}$$

où $\mathcal{P}_0$ représente la puissance du bruit environnant, et $\frac{1}{\theta}$ est le seuil critique à respecter. Comme évoqué plus haut, les technologies d'aujourd'hui tolèrent un seuil supérieur ou égal à 3 dB, ce qui correspond à $\frac{1}{\theta} = 2$.

Si le canal $c(i)$ s'avère être un candidat, alors - contrairement aux méthodes décentralisées classiques reposant sur une mesure local du SINR - le contrôleur centralisé vérifie ensuite que son utilisation par l'usager i ne dégrade pas les appels en cours pour ceux qui l'utilisent déjà. Précisément, la relation suivante doit être respectée :

$$\frac{P_{jq}}{\sum_{k \neq j | c(k)=c(i)} P_{kq} + P_{iq} + \mathcal{P}_0} \geq \frac{1}{\theta} \quad \forall\, j \mid c(j) = c(i), \tag{2.3}$$

où q représente la station avec laquelle l'usager j communique. Ceci garantit qu'aucun appel en cours ne sera interrompu.

De façon plus synthétique, en combinant les équations (2.2) et (2.3), une affectation est valide si et seulement si, pour tout usager i en liaison avec sa station p via le canal $c(i)$, on a :

$$\sum_{j \neq i | c(j)=c(i)} P_{jp} + \mathcal{P}_0 \leq \theta P_{ip}. \tag{2.4}$$

Le problème d'optimisation que l'on souhaite résoudre peut alors se formuler comme suit : comment assigner les canaux à un maximum d'usagers en respectant la contrainte (2.4) ?

La section suivante propose de découvrir les élements de théorie des graphes utilisés dans cette thèse pour traiter ce problème.

2.4 La théorie des graphes mise à contribution

La théorie des graphes est une branche des mathématiques discrètes dont l'avènement remonte au célèbre problème des ponts de Königsberg (voir Figure 2.6) publié par le célèbre mathématicien suisse Leonhard Euler en 1741. Ce n'est que près d'un siècle plus tard que de nouveaux problèmes susceptibles d'être résolus par les graphes font leur apparition : en 1847, le physicien allemand Gustav Kirchhoff énonça les lois portant maintenant son nom qui permettent de déterminer, dans un circuit électrique, l'intensité et le potentiel en différents points. Dix ans plus tard, le britannique Arthur Cayley s'intéressa aux graphes (plus précisément à la famille des graphes sans cycle, les arbres) pour énumérer les différentes façons d'agencer des atomes de carbone et d'hydrogène dans l'espace pour constituer une molécule appartenant à la famille des alcanes. Aujourd'hui, la théorie des graphes est une discipline très féconde ; les domaines liés à la notion de réseau tels que les transports ou les télécommunications sont des exemples parmi de nombreux autres possibles qui se sont développés notamment grâce à cette branche des mathématiques.

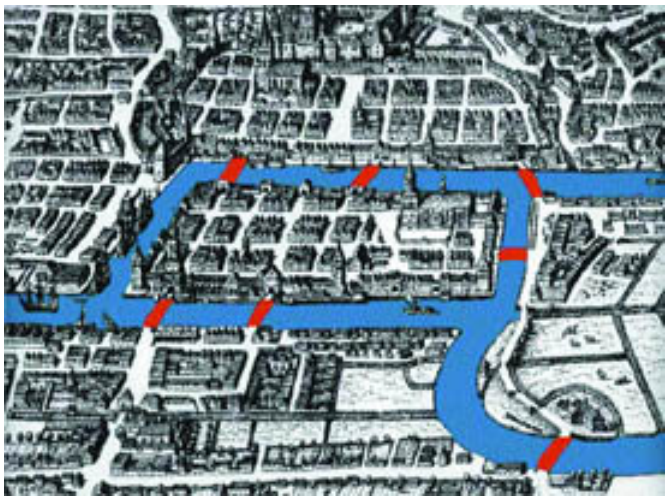


Figure 2.6 La Ville de Königsberg, sur la Pregel, était pourvue de 7 ponts et la question était de savoir si l'on pouvait imaginer une promenade dans la ville qui emprunterait chacun des 7 ponts une fois et une seule pour revenir à son point de départ. [4]

4. Source : domaine public.

2.4.1 Graphes et colorations

Définition 1 (Graphe)**.** *Un graphe est un couple (V, E), où V représente un ensemble de points (ou sommets), et E l'ensemble des arêtes reliant deux sommets de V.*

Par convention, on note $n = |V|$ la cardinalité de l'ensemble des sommets, et $m = |E|$ celle de l'ensemble des arêtes. Deux sommets u, v sont dits adjacents (ou voisins) si l'arête $\{u, v\} \in E$, et on note N_v l'ensemble des voisins d'un sommet v.

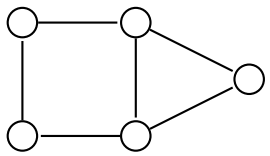


Figure 2.7 Un exemple de graphe

Ce qui importe avant tout c'est l'existence ou non d'un lien entre les sommets, et non les positions des sommets et des arêtes dans l'espace. On peut toutefois munir les arêtes ou les sommets de pondérations pour quantifier ce lien, par exemple pour indiquer la distance entre deux sommets. On peut également orienter les arêtes du graphe (celles-ci deviennent des arcs) pour préciser le sens de la relation binaire entre deux sommets. Par convention, l'ensemble des arcs se note alors A. Un graphe orienté et pondéré porte souvent le nom de réseau.

Un graphe est donc une représentation des liens entre les paires d'un ensemble de sommets. Il arrive souvent que l'on veuille caractériser certains sommets par un attribut commun, ou au contraire que l'on veuille imposer qu'ils aient un attribut distinct. Par exemple, supposons que l'on veuille colorer une carte géographique de sorte que deux pays voisins n'aient pas la même couleur. En termes de graphes, on peut représenter chaque pays par un sommet, et relier les sommets entre eux par une arête si les pays correspondants ont une frontière commune. Colorer la carte géographique revient alors à colorer les sommets du graphe de sorte que chaque arête ait ses deux extrémités de couleur différente. Cette notion de coloration des sommets a été formalisée en 1852 par Francis Guthrie, qui s'intéressait justement à la coloration de la carte des régions d'Angleterre. Il émis une conjecture aujourd'hui bien connue : le théorème des quatre couleurs [38], qui stipule que l'on peut toujours colorer une carte géographique avec 4 couleurs. Étonnamment, ce théorème n'a été prouvé qu'en 1976 ; la preuve exige l'usage d'un ordinateur et a longtemps fait débat au sein de la communauté scientifique. La coloration de graphes a été popularisée entre autres par ce problème et constitue aujourd'hui une branche à part entière de la théorie des graphes utilisée pour modéliser de nombreux problèmes d'optimisation combinatoire, tels que la planification d'examens [39] ou de tournois sportifs [40], mais aussi l'affectation de fréquences de téléphonie mobile (pour les modèles statiques) [8].

Définition 2 (k-coloration)**.** *Une k-coloration des sommets d'un graphe $G = (V, E)$ est une application $c : V \to \{1, \cdots, k\} \subset \mathbb{N}$, telle que*

$$c(u) = c(v) \quad \Rightarrow \quad \{u, v\} \notin E,$$

ce qui signifie que les extremités de chaque arête doivent être affectées d'une couleur différente. Un exemple est illustré à la Figure 2.8.

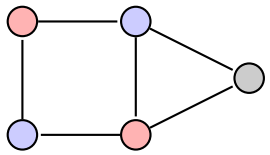


Figure 2.8 Un exemple de 3-coloration

Définition 3 (Nombre chromatique)**.** *Le nombre chromatique d'un graphe $G = (V, E)$, noté $\chi(G)$, est le plus petit nombre de couleurs nécessaires pour colorer tous les sommets du graphe :*

$$\chi(G) = \min\{k \in \mathbb{N} \mid \textit{il existe une } k\textit{-coloration } c : V \to \{1, \cdots, k\}\}.$$

Dans l'exemple précédent, il est clair que $\chi(G) = 3$, puisqu'il faut au moins trois couleurs pour colorer le triangle. Si l'on ne dispose que d'une seule ressource, ou couleur, un autre problème se pose naturellement : celui de déterminer le nombre maximum de sommets que l'on peut colorer à l'aide de cette couleur. Cet ensemble de sommets sera vraisemblablement un sous-ensemble de V, et motive les définitions suivantes.

Définition 4 (Sous-graphe induit)**.** *Soit un graphe $G = (V, E)$. Le graphe $G' = (V', E')$ est un sous-graphe partiel de G si $V' \subseteq V$, $E' \subseteq E$ et si les arêtes de E' ont bien leurs extrémités dans V'. De plus, si G' contient toutes les arêtes de G reliant deux sommets de V', alors G' est le sous-graphe induit par V'.*

Définition 5 (Stable)**.** *Un stable (ou ensemble indépendant) d'un graphe est l'ensemble des sommets du sous-graphe induit par une 1-coloration de ce graphe. Autrement dit c'est un ensemble de sommets deux à deux non adjacentes. La cardinalité du plus grand stable d'un graphe est notée $\alpha(G)$.*

Définition 6 (k-stable)**.** *Plus généralement, un k-stable d'un graphe est l'ensemble des sommets du sous-graphe induit par une k-coloration d'un sous-ensemble des sommets de G. La cardinalité du plus grand k-stable d'un graphe est notée $\alpha_k(G)$.*

La Figure 2.9 illustre un exemple de 2-stable (la couleur blanche n'en est pas une : le sommet à droite n'est pas coloré). Il est clair que dans cet exemple, on ne peut pas trouver de sous-graphe induit qui soit plus grand, puisqu'il faut au moins 3 couleurs pour colorer le triangle. Ce 2-stable a donc pour cardinalité $\alpha_2(G)$.

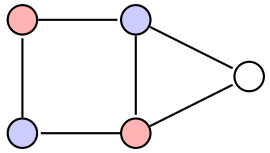


Figure 2.9 Un exemple de 2-stable

Notons que pour tout graphe $G = (V, E)$,

$$\alpha_k(G) \leq n \leq \chi(G) \cdot \alpha(G).$$

En effet, le plus grand sous-graphe induit k-colorable ne peut être plus grand que le graphe lui-même, et une k-coloration de G en $\chi(G)$ couleurs est une partition des sommets en $\chi(G)$ stables, de taille inférieure ou égale à $\alpha(G)$. De plus, il n'est pas difficile de montrer que

$$\chi(G) \leq k + n - \alpha_k(G),$$

puisque une k-coloration parmi d'autres consiste à colorer un k-stable en k couleurs, et les $n - \alpha_k(G)$ autres sommets restants de couleurs différentes.

Dans certaines applications, il est permis qu'un sommet ait des voisins de même couleur, à condition que cela ne se produise pas trop souvent. Ceci est formalisé à l'aide de la définition suivante.

Définition 7 (k-coloration θ-impropre)**.** *Si une arête $\{u, v\}$ vérifie $c(u) = c(v)$, on dit qu'un conflit a lieu pour les sommets u et v, et la coloration est dite impropre. Plus généralement, si au plus θ conflits ont lieu pour chaque sommet, la coloration est dite θ-impropre. Le nombre minimum de couleurs nécessaires pour obtenir une telle coloration est noté $\chi_\theta(G)$.*

La coloration θ-impropre est une généralisation de la coloration classique sans conflits (ou coloration propre) et a été introduite en premier par Lovász [41] en 1966 qui a démontré les premiers résultats sur le nombre chromatique θ-impropre $\chi_\theta(G)$. Ce n'est que vingt ans plus tard qu' Andrews et Jacobson [42], Harary et Fraughnaugh [43], et Cowen, Cowen et Woodall [44] ont remis cette théorie au goût du jour, de manière indépendante. Andrews et Jacobsen ont dérivé des bornes inférieures sur le nombre chromatique θ-impropre d'un

graphe, tandis que Haray et Fraughnaugh ont étudié ce paramètre comme un cas particulier d'une généralisation plus large du nombre chromatique. Enfin Cowen, Cowen et Woodall ont trouvé des bornes supérieures sur ce paramètre en vue de généraliser le fameux théorème des quatre couleurs [45] évoqué plus haut.

Puisque une coloration θ-impropre est également $(\theta+1)$-impropre, il est clair que $\chi_\theta(G) \geq \chi_{\theta+1}(G)$. De plus, puisqu'une classe de couleur d'une coloration θ-impropre induit un sous-graphe dont les sommets ont au plus θ voisins, il est toujours possible de séparer cette classe en $\theta+1$ ensembles indépendants (nous reviendrons sur ce point dans la section suivante), d'où les inégalités

$$\left\lceil \frac{\chi(G)}{\theta+1} \right\rceil \leq \chi_\theta(G) \leq \chi(G).$$

Très récemment (en 2010), Araujo et al. [46] ont introduit la notion de k-coloration θ-impropre *pondérée* pour traiter un problème d'affectation de fréquences de satellites. Celle-ci se définit de la façon suivante.

Définition 8 (k-coloration θ-impropre pondérée)**.** *Soit un graphe $G=(V,E)$, ω une application dans $\mathbb{R}_+^*$ qui pondère les arêtes de G, et θ un réel positif. Une coloration θ-impropre pondérée par ω est une coloration qui vérifie*

$$\sum_{u \in N_v | c(v)=c(u)} \omega(\{u,v\}) \leq \theta \quad \forall\, v \in V,$$

ce qui signifie qu'un sommet peut être en conflit avec ses voisins, pourvu que la somme des poids des arêtes le reliant à ces sommets soit inférieure à θ. Le nombre minimum de couleurs nécessaires pour obtenir une telle coloration est noté $\chi_\theta(G,\omega)$. Un ensemble de sommets affectés de la même couleur vérifiant cette inégalité est appelé ensemble θ-indépendant.

Un exemple de 2-coloration 2-impropre est représenté à la Figure 2.10 ci-après : malgré le fait qu'il existe une arête entre les sommets a et b, le poids $\omega(\{a,b\}) = 1$ est inférieur à $\theta = 2$, donc le conflit est permis. En revanche, si l'on colore le sommet c de la même couleur, la somme des poids des arêtes incidentes à c ($1.5 + 1.5 = 3$) excède le seuil admissible : une autre couleur est donc nécessaire.

Notons que si les poids sont unitaires et que θ est entier, on retombe sur la définition classique des colorations impropres. De même, quelque soient les pondérations, si θ est égal à 0, on retombe sur un problème de coloration propre.

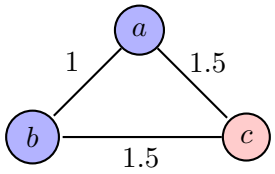


Figure 2.10 Un exemple de 2-coloration 2-impropre

Plus grand sous-graphe induit et k-coloration θ-impropre pondérée constituent les ingrédients de modélisation principaux de notre problème d'affectation de canaux. Voyons ce qu'il en est précisément.

2.4.2 Modélisation du problème en termes de graphe

Reprenons le problème d'affectation introduit à la section précédente. Considérons un ensemble d'usagers $\{1, \cdots, n\}$ en attente d'un canal souhaitant se connecter au réseau, et créons un graphe G qui a pour ensemble de sommets l'ensemble de ces n usagers. L'usager i est à proximité d'une station $a(i)$ et émet une puissance égale à $P_{ia(i)}$ vers cette station. Associons ainsi un poids $W(i)$ à chaque sommet, égal à la puissance du signal émis $P_{ia(i)}$. La propagation des ondes étant omni-directionnelle, l'usager i émet également un signal vers toutes les autres stations. Ce signal est perçu comme une interférence pour tout usager j, connecté à la station $a(j)$. De même, la station $a(i)$ reçoit les signaux de tous les autres usagers, qui sont perçus comme des interférences pour l'usager i. Rajoutons donc deux arcs entre toute paire de sommets i et j : l'arc (i, j) de poids $P_{ia(j)}$ et l'arc (j, i) de poids $P_{ja(i)}$. Un exemple de sous-graphe induit par deux sommets est illustré à la Figure 2.11, à droite.

On obtient un graphe complet, orienté, dont les sommets et les arcs sont pondérés. En poursuivant la logique de Araujo et al. [46], on généralise la notion de coloration θ-impropre à ce graphe en imposant

$$\sum_{u \neq v | c(v)=c(u)} \omega(u, v) \leq \theta W(v) \quad \forall\, v \in V. \tag{2.5}$$

En interprétant les canaux comme des couleurs, le lien entre une telle coloration et les contraintes de respect du rapport signal-bruit s'éclaircit :

$$\sum_{j \neq i | c(j)=c(i)} P_{jp} + \mathcal{P}_0 \leq \theta P_{ip} \quad \Leftrightarrow \quad \sum_{j \neq i | c(j)=c(i)} w(j, i) + \mathcal{P}_0 \leq \theta W(i).$$

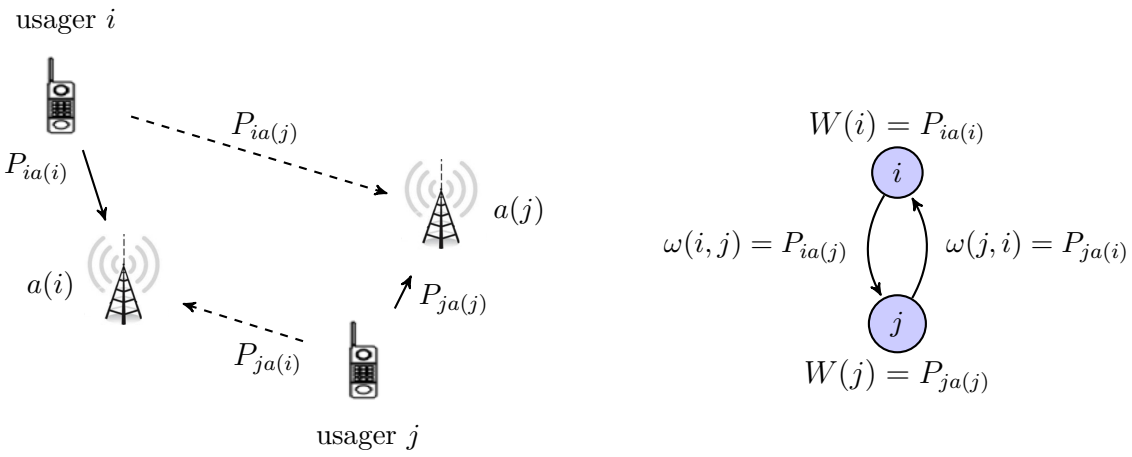


Figure 2.11 Modélisation du problème d'affectation en un problème de graphe

Trois remarques importantes découlent de cette transformation. Premièrement, ce graphe modélise des configurations dites en liaison montante, ce qui signifie que l'on s'intéresse aux signaux émis par les usagers, reçus par les stations de base. Pour obtenir le modèle en liaison descendante, il suffit d'inverser le sens des arcs du graphe. Deuxièmement, le bruit ambiant $\mathcal{P}_0$ est une constante que l'on peut prendre en compte dans le graphe en créant k sommets artificiels $x_1, \cdots, x_k$ de poids $W(x_i) = 0$, chacun coloré d'une couleur parmi les k disponibles, ayant des arcs depuis et vers chaque noeud $u \in V \setminus \{x_1, \cdots, x_k\}$ tels que $\omega(x_i, u) = \mathcal{P}_0$, $\omega(u, x_i) = 0$, et des arcs entre chaque paire de sommets artificiels munis de n'importe quel poids strictement positif. Enfin, tel que mentionné en introduction, les technologies de multiplexage des systèmes de téléphonie mobile modernes divisent les canaux en sous-canaux. Ceci peut également être pris en compte en multipliant le nombre de couleurs k par le nombre de sous-canaux. Ces remarques montrent qu'il n'y a aucune perte de généralité lors de la transformation en un problème de graphe.

Maximiser le nombre d'usagers connectés au réseau via un canal est alors équivalent à maximiser le nombre de sommets colorés, au sens de de l'équation (2.5). Il s'agit donc de trouver le plus grand sous-graphe induit qui soit k-colorable de façon θ-impropre. Dorénavant, la cardinalité de la solution optimale sera notée $\alpha_k^\theta(G, W, \omega)$. Un exemple est représenté à la Figure 2.12 ci-après. Si l'on dispose d'une seule couleur ($k = 1$) et que $\theta = \frac{1}{2}$, alors $\alpha_k^\theta(G, W, \omega) = 2$. En effet, si l'on colore les sommets b et c, la somme des poids des arcs entrant en b sera au moins égale à 3, ce qui excède $\theta\, W(b) = \frac{5}{2}$.

La Figure 2.13 ci-après illustre un exemple récapitulatif des différentes étapes du processus d'affectation. Un graphe complet, pondéré, orienté est créé sur les usagers (pour alléger le dessin, les arcs sont remplacés par des arêtes, et les poids ne sont pas représentés), et une

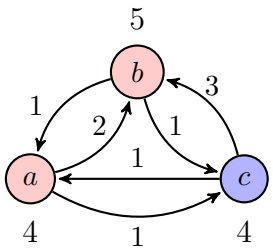


Figure 2.12 Un exemple, $\alpha_1^{\frac{1}{2}}(G, W, \omega) = 2$

coloration θ-impropre des sommets correspond à une affectation valide de canaux.

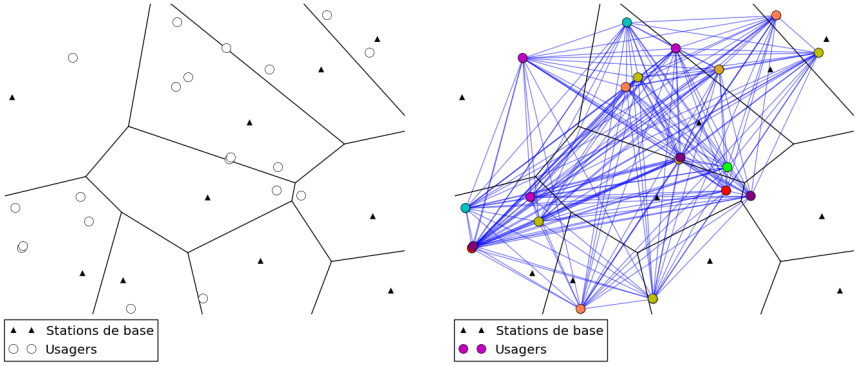


Figure 2.13 Un exemple avec 10 stations de base et 20 usagers répartis aléatoirement dans le plan

Unicité du modèle

Bien que l'on trouve des modèles semblables dans la littérature, ce modèle est bel et bien unique à notre connaissance. On peut mentionner au moins deux modèles similaires de coloration θ-impropres sur des graphes orientés et pondérés. En 1991, une équipe de chercheurs japonais [47] a formalisé pour la première fois le lien entre l'affectation de canaux et la coloration impropre. Les graphes utilisés sont similaires à ceux utilisés ici, mais diffèrent essentiellement par la structure de coûts sur les arcs et les sommets. Tout d'abord, les pondérations sont normalisées en divisant $\omega(u, v)$ par $W(v)$, de façon à avoir des poids unitaires sur les sommets. Si deux sommets u et v sont suffisamment éloignés, les arcs (u, v) et (v, u) sont retirés du graphe (ceci revient à fixer leur poids à 0). Si u et v sont affectés à la même station de base, les poids $\omega(u, v)/W(v)$ et $\omega(v, u)/W(u)$ prennent n'importe quelle valeur strictement plus grande que θ. Sinon, le poids prend une valeur qui dépend des distances entre les

stations de base de u et v. En d'autres mots, ils utilisent une structure de coût plus grossière que celle proposée ici, en s'appuyant uniquement sur des considérations géométriques. Ils ont montré que ce modèle permettait d'avoir une plus grande efficacité spectrale que les modèles statiques classiques et ont étudié le paramètre $\chi_\theta(G, W, \omega)$, le nombre minimum de couleurs nécessaires pour obtenir une coloration θ-impropre. Curieusement, ils n'ont pas cité les travaux de Lovász [41] et ont redémontré certains de ses résultats publiés 25 ans plus tôt. D'autre part ils n'ont plus été cités dans aucun des travaux similaires apparus près de 20 ans plus tard [3, 46, 48]. Archetti et al. [3] utilisent un graphe semblable à celui de Tamura et al. [47], mais affinent la structure de coûts : les poids sur les arcs et sur les sommets correspondent aux puissances réelles, ce qui permet de tenir compte de la configuration et de la position des antennes, mais aussi de l'affaiblissement de propagation (voir le paragraphe 2.1). Cela donne lieu à des graphes complets (contrairement aux premiers). De plus, Archetti et al. ont considéré une fonction objectif supplémentaire : en fixant k, le nombre de canaux disponibles, ils ont minimisé θ de façon à garantir un minimum d'interférences sur l'ensemble du réseau. Le modèle proposé ici prolonge cette aspect, puisque nous proposons de fixer k et θ et de maximiser le nombre d'usagers affectés d'un canal.

D'autres modèles s'appuyant sur des graphes non orientés méritent également d'être mentionnés. Premièrement, Araujo et al. [46] ont modélisé les affectations de fréquences radio dans un réseau de satellites à l'aide d'un graphe similaire mais qui n'est ni orienté, ni complet, et où les sommets ne sont pas pondérés. Tout comme Archetti et al., ils se sont intéressés à la minimisation des paramètres $\chi_\theta(G, \omega)$ et de θ. Enfin, Mishra et al. [48] ont étudié l'affectation de ressources dans les réseaux locaux sans fil (WLAN) à l'aide d'un graphe complet, avec un sommet par usager, où toutes les arêtes sont munies d'un poids $\omega(\{i, j\}) = P_{jp} + P_{iq}$, où p et q représentent les points d'accès du réseau pour les usagers i et j, respectivement. Ils ont alors minimisé les interférences sur l'ensemble du réseau en cherchant une coloration impropre qui minimise $\sum_{\{i,j\} \in E | c(i)=c(j)} \omega(\{i, j\})$.

L'originalité du modèle proposé ici repose sur l'utilisation de colorations impropres sur des graphes orientés, complets et pondérés, et celui-ci est unique de par la fonction objectif étudiée, dont le maximum est égal au paramètre $\alpha_k^\theta(G, W, \omega)$, la taille du plus grand sous-graphe induit colorable de façon θ-impropre, ou en termes de télécommunications, le plus grand nombre d'usagers étant connectés simultanément au réseau.

Dans la section suivante, nous allons montrer en quoi il s'agit d'un problème difficile au sens combinatoire.

2.4.3 Complexité

L'explosion combinatoire

Curieusement, de nombreux problèmes de théorie des graphes s'énoncent de façon très simple, mais sont très difficiles à résoudre, dans la mesure où l'usage d'un ordinateur est presque systématiquement indispensable ; le problème de coloration d'une carte géographique au moyen de quatre couleurs en est un parfait exemple. Il n'est donc pas étonnant que l'algorithmique soit intimement liée à la théorie des graphes. L'intensité des recherches en théorie des graphes est due en partie au fait que de nombreux problèmes sont si complexes qu'ils « résistent » à tout algorithme et sont intrinsèquement difficiles ; cette notion de difficulté d'un problème combinatoire a été formalisée par Cook et Karp [49] qui ont baptisé cette classe de problèmes comme étant $\mathcal{NP}$-difficiles. À ce jour, ces problèmes ne peuvent pas être résolus de façon exacte, sans que le temps de résolution de l'algorithme utilisé n'augmente de façon exponentielle avec la taille du problème (le nombre de sommets du graphe par exemple) : on parle *d'explosion combinatoire*. Pour comprendre l'ampleur de cette difficulté, prenons l'exemple du problème de minimisation des couleurs nécessaires pour colorer les sommets d'un graphe (i.e., la recherche du paramètre χ). Si l'on voulait examiner toutes les affectations possibles, il faudrait, pour chacun des n sommets, considérer les n couleurs possibles, soit un total de n^n solutions candidates (bien que nombre d'entre elles soient symétriques). Pour un graphe à 50 sommets, cela donne environ $8,8 \times 10^{84}$ possibilités, soit plus que le nombre d'atomes de l'univers [50] ! Aujourd'hui, on peut résoudre de façon exacte le problème en quelques secondes si $n = 80$, en moins d'une journée si $n = 90$, mais au delà, cela devient hors de notre portée. De nombreux problèmes combinatoires appartiennent à cette classe de problèmes $\mathcal{NP}$-difficiles, on peut en consulter une liste dans [51].

Définition 9 (Complexité)**.** *La complexité, ou le temps d'exécution, d'un programme P est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc.) nécessaires à l'exécution de P. Lorsque cette complexité dépend d'un paramètre n, on dira que P a une complexité en $\mathcal{O}(f(n))$, s'il existe $K > 0$ tel que la complexité de P est au plus $Kf(n)$, pour tout n suffisamment grand. En particulier, si $f(n)$ est une fonction polynomiale, on dit que P s'exécute en temps polynomial.*

Définition 10 (Classe de problèmes $\mathcal{P}$)**.** *Un problème de décision (dont la réponse est oui ou non) appartient à la classe $\mathcal{P}$ s'il peut être résolu en temps polynomial.*

Définition 11 (Classe de problèmes $\mathcal{NP}$)**.** *Un problème de décision appartient à la classe $\mathcal{NP}$ s'il est possible de vérifier à l'aide d'une machine de Turing non déterministe si la réponse est oui en temps polynomial.*

Il est clair que $\mathcal{P} \subseteq \mathcal{NP}$. En effet, si l'on dispose d'un algorithme pouvant résoudre un problème de $\mathcal{P}$ en temps polynomial, il suffit de l'exécuter pour vérifier qu'une solution est bien valide. L'inclusion inverse n'est pas triviale du tout, et fait même l'objet d'un problème considéré par l'Institut Clay comme l'un des plus importants du millénaire. Il est possible de montrer que si l'inclusion inverse est fausse, alors on ne peut espérer résoudre en temps polynomial tout problème de décision difficile au sens combinatoire. Bien que la communauté scientifique s'accorde sur le fait que $\mathcal{P} \neq \mathcal{NP}$, il n'en existe à ce jour aucune preuve.

Définition 12 (Problème $\mathcal{NP}$-difficile)**.** *En supposant que $\mathcal{P} \neq \mathcal{NP}$, un problème de décision est $\mathcal{NP}$-complet s'il n'existe aucun algorithme pouvant s'exécuter en temps polynomial pour le résoudre. La version non décisionnelle d'un tel problème est alors dite $\mathcal{NP}$-difficile.*

Il est donc indispensable de se poser la question de savoir si la recherche du paramètre $\alpha_k^\theta(G, W, \omega)$ appartient à cette classe de problèmes, auquel cas il n'est pas pertinent de chercher à développer des algorithmes exacts et polynomiaux. La proposition suivante clôt le débat (à moins que $\mathcal{P} = \mathcal{NP}$) : il s'agit bel et bien d'un problème $\mathcal{NP}$-difficile.

Lemme 1. *Le problème qui consiste à déterminer $\alpha_k(G)$ est $\mathcal{NP}$-difficile.*

Démonstration. Le problème de décision associé, qui consiste à déterminer s'il est possible de colorer un nombre donné de sommets $U \subseteq V$ avec k couleurs, appartient clairement à la classe des problèmes $\mathcal{NP}$. En effet, il suffit de parcourir en un temps $\mathcal{O}(m)$ l'ensemble des arêtes du sous-graphe induit par U pour vérifier qu'il s'agit bien d'une coloration. De plus, en fixant ce nombre à n, cela est équivalent à déterminer si le graphe est entièrement k-colorable. Par conséquent déterminer $\alpha_k(G)$ est un problème plus difficile que déterminer si le graphe est k-colorable, qui est un problème $\mathcal{NP}$-complet [52]. Il s'ensuit que déterminer $\alpha_k(G)$ est $\mathcal{NP}$-difficile. □

Ceci n'est pas étonnant car si l'on fixe k à 1, on retrouve le problème du stable maximum, qui est déjà un problème combinatoire difficile.

Proposition 1. *Le problème qui consiste à déterminer $\alpha_k^\theta(G, W, \omega)$ est $\mathcal{NP}$-difficile.*

Démonstration. Le problème de décision associé, qui consiste à déterminer s'il est possible de colorer θ-improprement un nombre donné de sommets $U \subseteq V$ avec k couleurs, appartient clairement à la classe des problèmes $\mathcal{NP}$. En effet, il suffit de parcourir en un temps $\mathcal{O}(n)$ l'ensemble U pour vérifier qu'il s'agit bien d'une coloration impropre. Et puisque déterminer $\alpha_k(G)$ est $\mathcal{NP}$-difficile d'après le Lemme 1, il suffit de montrer que ce problème se réduit au problème de déterminer $\alpha_k^\theta(G, W, \omega)$ en un nombre polynomial d'étapes. Pour ce faire, il

suffit de compléter G de façon à obtenir un graphe complet, et de remplacer chaque arête $\{u, v\}$ par deux arcs (v, u) et (u, v) de poids 2 si $\{u, v\} \in E$, et 0 sinon. On munit ensuite tous les sommets d'un poids unitaire, et l'on fixe le seuil θ à 1. Alors, il existe une k-coloration θ-impropre de U si et seulement s'il existe une affectation des couleurs aux sommets de U telle que

$$\sum_{u \neq v | c(u)=c(v)} \omega(u, v) \leq \theta W(v) = 1 \quad \forall v \in U,$$

ce qui n'est possible que si et seulement si $\omega(u, v) = 0$ pour toute paire (u, v) de sommets de U telle que $c(u) = c(v)$, autrement dit que si et seulement si ces paires n'appartiennent pas l'ensemble des arêtes de G, et donc si et seulement s'il s'agit d'une coloration des sommets de U. □

Résultats complémentaires

Pour montrer que la recherche de $\alpha_k(G)$ est un problème $\mathcal{NP}$-difficile, nous avons montré que le problème de k-colorabilité d'un graphe peut se réduire en un nombre polynomial d'opérations au problème du plus grand sous-graphe induit k-colorable. D'autres réductions sont possibles, par exemple via le problème du stable maximum. Narasimhan [53] a détaillé cette réduction, mais a également montré la réduction inverse : si l'on sait résoudre le problème du stable maximum, alors on peut déterminer $\alpha_k(G)$ moyennant une transformation comportant un nombre polynomial d'opérations élémentaires, appelée produit cartésien entre deux graphes [54].

Définition 13 (Produit cartésien)**.** *Soient $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$ deux graphes. Le graphe $G_1 + G_2$ est le graphe $G = (V, E)$ tel que*

$$\begin{cases} V = \{(x, y) \mid x \in V_1,\ y \in V_2\} \\ E = \{\{(x, y), (z, w)\} \mid x = z,\ \{y, w\} \in E_2 \text{ ou } y = w,\ \{x, z\} \in E_1\} \end{cases}$$

Narasimhan [53] a montré que pour déterminer $\alpha_k(G)$, il suffit d'effectuer le produit cartésien de $G = (V, E)$ avec un graphe complet d'ordre k, et de déterminer $\alpha(G)$ dans le graphe résultant. Par exemple, dans la Figure 2.14, on cherche deux ensembles stables de taille totale $\alpha_2(G)$, ce qui revient à chercher le plus grand stable dans le graphe $G + K_2$. Dans cet exemple trivial, on a $\alpha_2(G) = \alpha(G + K_2) = 2$ (avec les sommets 1 et 0 d'une part, et les sommets 04 et 15 d'autre part).

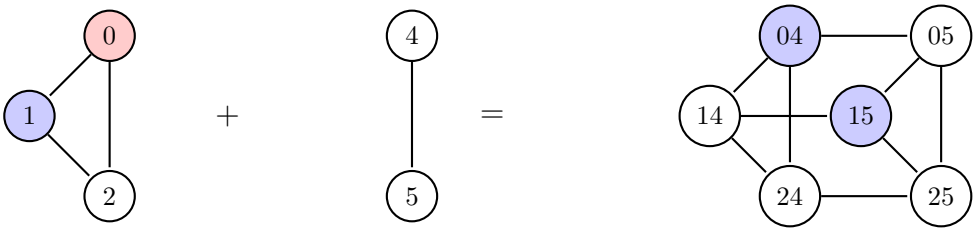


Figure 2.14 Un exemple : $\alpha_2(G) = \alpha(G + K_2)$

Nous avons donc affaire à un problème difficile au sens combinatoire, dans la mesure où il n'existe a priori aucun algorithme pouvant trouver la solution optimale en un temps raisonnable pour de grandes instances.

La section qui suit passe en revue les différentes stratégies possibles pour faire face à ce type de difficulté.

2.5 Comment traiter un problème $\mathcal{NP}$-difficile ?

2.5.1 Algorithmes exacts

Il existe au moins quatre voies possibles pour traiter un problème appartenant à la classe des problèmes $\mathcal{NP}$-difficiles. Tout d'abord, si la taille des instances n'est « pas trop élevée » (cette notion est bien sûr arbitraire, disons si $n \leq 100$ pour la recherche du nombre chromatique), il est possible d'utiliser des algorithmes exacts, bien que non polynomiaux. La programmation linéaire en nombres entiers, plus particulièrement les algorithmes dits de séparation et évaluation et de plans coupants, sont les approches privilégiées dans ce cas de figure. Le livre de Wolsey [55] est une bonne introduction à ces concepts.

Lorsque les contraintes du problème sont linéaires, elles forment un polyèdre dans un espace de dimension égale au nombre de variables, et lorsque l'objectif est aussi linéaire, la solution optimale se trouve sur un point extrême ou « coin » du polyèdre et peut être obtenue en temps polynomial. L'algorithme du simplexe inventé par Dantzig [56] consiste à parcourir itérativement les points extrêmes du polyèdre ; c'est une façon très rapide d'obtenir la solution optimale (bien que sa complexité ne soit pas polynomiale, nous reviendrons sur ce point).

Lorsque les variables sont contraintes à être entières, le problème se corse car la solution optimale ne se trouve plus forcément sur un point extrême. Les algorithmes de séparation et évaluation (ou branch-and-bound), basés sur la stratégie *diviser pour régner*, consistent à partitionner le polyèdre itérativement en sous-polyèdres, dans l'espoir que des solutions

entières se trouvent sur un point extrême d'un des sous-polyèdres. La difficulté réside dans le fait qu'il est difficile de savoir quelle partition du polyèdre est la plus prometteuse, et puisqu'il en existe un nombre exponentiel, l'explosion combinatoire est bien inhérente à cette méthode. Quant à eux, les algorithmes de plans coupants consistent à redéfinir itérativement le polyèdre, jusqu'à ce qu'un point extrême corresponde à une solution entière. La difficulté est ici de savoir comment redéfinir le polyèdre à chaque itération, de façon à éliminer les régions inutiles. De même, le nombre d'itérations nécessaires pour faire cela sur un polyèdre quelconque peut être exponentiel [57]. Typiquement ces deux méthodes sont utilisées de façon conjointe et donnent lieu à un algorithme dit de branch-and-cut.

Dans tous les cas, de telles méthodes nécessitent de mettre en équations le problème, c'est-à-dire de définir les variables, l'objectif et les contraintes. Des solveurs commerciaux (ou en open source) tels que `Cplex` ou `Gurobi` permettent alors la résolution de ces modèles (implanter soi-même ces algorithmes est une tâche ardue).

Modèles linéaires en nombres entiers pour $\chi_\theta(G, \omega)$

Araujo et al. [46] ont cherché à déterminer le paramètre $\chi_\theta(G, \omega)$ de cette façon sur des graphes non orientés, uniquement pondérés sur les arêtes. Voici le modèle utilisé : soit x_{vi} et y_i des variables binaires, où x_{vi} prend la valeur 1 si et seulement si $v \in V$ prend la couleur $i \in \{1, ..., n\}$, et où y_i prend la valeur 1 si et seulement si la couleur $i \in \{1, ..., n\}$ est utilisée. Le modèle prend alors la forme suivante :

$$\min_{x,y} \sum_{i=1}^{n} y_i \tag{2.6}$$

sujet à :

$$\sum_{i=1}^{n} x_{vi} = 1 \qquad \forall v \in V \tag{2.7}$$

$$x_{vi} \leq y_i \qquad \forall v \in V, \forall i \in \{1, \cdots, n\} \tag{2.8}$$

$$\sum_{u \in N_v} \omega(\{u, v\}) \cdot x_{ui} \leq \theta + M_v \cdot (1 - x_{vi}) \qquad \forall v \in V, \forall i \in \{1, \cdots, n\} \tag{2.9}$$

$$x_{vi},\ y_i \in \{0, 1\} \qquad \forall v \in V, \forall i \in \{1, \cdots, n\} \tag{2.10}$$

La fonction objectif (2.6) vise à minimiser le nombre de couleurs utilisées. Les contraintes (2.7) assurent l'affectation d'une et une seule couleur à chaque sommet. Les contraintes (2.8) lient

les variables x_{vi} et y_i, de sorte que l'objectif augmente d'une unité des lors qu'une nouvelle couleur est utilisée. Les inéquations (2.9) sont des contraintes disjonctives qui imposent que la coloration obtenue soit bien θ-impropre ; M_v est une constante assez grande, par exemple $M_v = \sum_{u \in N_v} \omega(u, v) - \theta$ suffit.

Ce modèle est alors résolu par un algorithme de séparation et évaluation, dont le temps de résolution est assez rapide, bien que non utilisable en pratique. Par exemple, sur un graphe à 2 000 sommets avec deux couleurs, après un temps d'exécution d'environ 600 secondes, l'algorithme de Araujo et al. présente un écart relatif d'au moins 50% avec la solution optimale. Ce modèle est néanmoins intéressant, car si l'on relâche les contraintes d'intégrité, on obtient une borne inférieure sur le paramètre $\chi_\theta(G, \omega)$.

Cas orienté

Il est facile d'adapter le modèle précédent pour les graphes orientés, il suffit dans la contrainte (2.9) de sommer sur les prédécesseurs de v et non sur ses voisins. Archetti et al. [3] ont proposé une autre formulation pour calculer $\chi_\theta(G, W, \omega)$ dont la relaxation linéaire est beaucoup plus serrée, et donnant donc une meilleure borne inférieure. Tel que décrit précédemment, la famille de graphes utilisés est une version asymétrique de celle de Araujo et al. [46], avec en plus des poids sur les sommets. Appelons Ω l'ensemble de tous les sous-ensembles θ-independants de G, et soit λ_p une variable binaire égale à 1 si et seulement si le sous-ensemble $p \in \Omega$ est selectionné. Le modèle proposé s'écrit alors comme suit :

$$\min_{\lambda} \sum_{p \in \Omega} \lambda_p \tag{2.11}$$

sujet à :

$$\sum_{p \in \Omega | v \in p} \lambda_p = 1 \qquad \forall v \in V \tag{2.12}$$

$$\lambda_p \in \{0, 1\} \qquad \forall p \in \Omega \tag{2.13}$$

Cette formulation, appelée décomposition de Dantzig-Wolfe, est alors résolue par un procédé de génération de colonnes : les colonnes ou variables λ_p sont générées dynamiquement à l'aide d'un sous-problème. Plus de détails sont données dans le Chapitre 3. Ce procédé, une fois intégré dans un arbre de branchement, donne lieu à un algorithme de *branch-and-price*. Archetti et al. résolvent des instances de graphes ayant jusqu' à 300 sommets, mais avec 15 couleurs ; dans le meilleur des cas cela nécessite de l'ordre de 300 secondes. Ceci dit, la relaxation linéaire est de très bonne qualité puisqu'une majorité des instances sont résolues au noeud racine.

Modèle linéaire en nombres entiers pour $\alpha_k(G)$

Pour calculer $\alpha_k(G)$, un premier modèle consiste à utiliser des variables binaires x_{vi} égales à 1 si et seulement le sommet v prend la couleur i parmi l'ensemble des k couleurs disponibles. La formulation s'écrit alors comme suit :

$$\max_x \sum_{i=1}^{k} \sum_{v \in V} x_{vi} \tag{2.14}$$

sujet à :

$$\sum_{i=1}^{n} x_{vi} \leq 1 \qquad \forall v \in V \tag{2.15}$$

$$x_{ui} + x_{vi} \leq 1 \qquad \forall \{u, v\} \in E, \forall i \in \{1, \cdots, k\} \tag{2.16}$$

$$x_{vi} \in \{0, 1\} \qquad \forall v \in V, \forall i \in \{1, \cdots, k\} \tag{2.17}$$

L'objectif (2.14) vise à maximiser le nombre de sommets colorés, les contraintes (2.15) imposent que chaque sommet ait au plus une couleur, et les contraintes (2.16) assurent que les arêtes aient leurs extrémités colorées différemment (si elles le sont). Ce dernier groupe de contraintes peut également s'écrire comme une contrainte disjonctive, que l'on obtient en sommant sur tous les sommets u de N_v :

$$\sum_{u \in N_v} (x_{ui} + x_{vi}) \leq |N_v| \quad \forall v \in V, \forall i \in \{1, \cdots, k\}.$$

En réordordonnant les termes, on retrouve la contrainte (2.9) de la formulation de Araujo et al. [46] avec $\theta = 0$ et $\omega(\{e\}) = 1$ pour toute arête e :

$$\sum_{u \in N_v} x_{ui} \leq |N_v| \cdot (1 - x_{vi}) \quad \forall v \in V, \forall i \in \{1, \cdots, k\}.$$

On trouve également dans les travaux de [58] une décomposition de Dantzig-Wolfe de ce modèle pour le cas particulier où $k = 1$.

Une approche similaire est proposée dans le Chapitre 3 pour déterminer $\alpha_k^\theta(G, W, \omega)$. Nous verrons plus en détail comment améliorer ces modèles, comment les résoudre, et comment les temps de résolution peuvent être diminués.

2.5.2 Bornes

Nous avons vu qu'en omettant les contraintes d'intégralité des modèles précédents, on obtient des problèmes linéaires (pouvant être résolus en temps polynomial), donnant des bornes de plus ou moins bonne qualité sur l'objectif. Une deuxième possibilité consiste à déterminer des bornes par des considérations logiques. Cette méthode est souvent utilisée de façon conjointe avec la programmation en nombre entiers.

Bornes sur le nombre chromatique

Par exemple, il est possible d'encadrer le nombre chromatique d'un graphe à l'aide de n et m de la façon suivante :

$$\left\lceil \frac{n^2}{n^2 - 2m} \right\rceil \leq \chi(G) \leq \min \left\{ n, \left\lfloor \sqrt{2m + \frac{1}{4}} + \frac{1}{2} \right\rfloor \right\}.$$

La borne inférieure est loin d'être triviale, une preuve par récurrence se trouve dans [59], mais la borne supérieure est plus aisée à saisir. D'une part il est évident que n couleurs sont suffisantes pour colorer n'importe quel graphe, d'autre part le nombre d'arêtes d'un graphe G est nécessairement supérieur à $\binom{\chi(G)}{2}$, sinon dans toute coloration optimale de G, il existerait deux classes de couleur reliées par aucune arête que l'on pourrait alors fusionner, contredisant le fait que la coloration est optimale. En isolant $\chi(G)$ dans cette inéquation, on trouve bien la borne supérieure ci-dessus. Ces bornes inférieure et supérieure sont bien sûr faciles à calculer et permettent de diminuer drastiquement les temps de résolution de programmes en nombres entiers comme en témoignent les tests numériques de [60]. Il est important de remarquer que ces bornes peuvent aussi bien être des égalités (par exemple pour un triangle), que des inégalités avec un écart arbitrairement grand (par exemple, une étoile à m branches et $n + 1$ sommets peut toujours être colorée en 2 couleurs, alors que $\min\{n, \left\lfloor \sqrt{2m + \frac{1}{4}} + \frac{1}{2} \right\rfloor\}$ peut être arbitrairement grand).

D'autres bornes particulièrement intéressantes existent ; de nouvelles définitions sont nécessaires pour les comprendre.

Définition 14 (Degré)**.** *Le degré* $\deg(v)$ *d'un sommet* v *d'un graphe* $G = (V, E)$ *correspond à son nombre de voisins :*

$$\deg(v) = |N_v|,$$

et le maximum des degrés parmi tous les sommets de V *est noté* $\Delta(G)$*.*

Si l'on colore les sommets séquentiellement, il est clair qu'un maximum de $\Delta(G)$ couleurs peuvent déjà apparaître dans le voisinage d'un sommet donné, et il s'ensuit donc que

$$\chi(G) \leq \Delta(G) + 1,$$

ce qui fournit une deuxième borne supérieure sur le nombre chromatique (ces bornes ne se dominent pas). Dans le même esprit, Lovász [41] a étendu ce résultat aux colorations θ-impropres :

$$\chi_\theta(G) \leq \left\lceil \frac{\Delta(G)+1}{\theta+1} \right\rceil,$$

qui elle-même a été généralisée par Araujo et al. [46] pour les graphes pondérés par une fonction $\omega : E \to \mathbb{R}_+^*$.

Définition 15 (Degré pondéré)**.** *Le degré pondéré d'un sommet v d'un graphe $G = (V, E, \omega)$ correspond à la somme des poids des arêtes incidentes à ce sommet :*

$$\deg(v, \omega) = \sum_{u \in N_v} \omega(\{u, v\}),$$

et le maximum des degrés pondérés parmi tous les sommets de V est noté $\Delta(G, \omega)$. Notons qu'avec des poids unitaires, on retombe sur la définition classique du degré.

On a alors la borne supérieure suivante, pour tout $\varepsilon > 0$:

$$\chi_\theta(G, \omega) \leq \left\lceil \frac{\Delta(G, \omega) + \varepsilon}{\theta + \gcd(\{\omega\})} \right\rceil,$$

où $\gcd(\{\omega\})$ est le plus grand commun diviseur (au sens des réels) de tous les poids $\omega(\{u, v\})$, et θ est un réel positif. On peut noter que pour des graphes complets munis de pondérations constantes et égales, cette borne est atteinte.

Cas orienté

Lorsque le graphe est orienté, on distingue le degré entrant $deg^-(v)$ du degré sortant $deg^+(v)$, qui sont respectivement le nombre de prédécesseurs et de successeurs de v. Ces définitions s'adaptent également pour un graphe orienté, pondéré par $\omega : A \to \mathbb{R}_+^*$ de la façon suivante.

Définition 16 (Degré entrant pondéré)**.** *Le degré entrant pondéré d'un sommet v d'un réseau $G = (V, A, \omega)$ correspond à la somme des poids des arcs entrant en v :*

$$\deg^-(v, \omega) = \sum_{u | (u,v) \in A} \omega(u, v),$$

et le maximum des degrés entrants pondérés parmi tous les sommets de V est noté $\Delta^-(G,\omega)$. Notons qu'avec des poids unitaires, on retombe sur la définition classique du degré entrant. La définition pour le degré sortant pondéré est analogue.

Très récemment, Bang-Jensen et al. [61] ont prouvé par des considérations algébriques une borne supérieure calculable en temps polynomial sur $\chi_1(G,\omega)$ pour les graphes orientés, en s'appuyant sur ce paramètre :

$$\chi_1(G,\omega) \leq \left\lfloor 2\Delta^-(G,\omega) + 1 \right\rfloor,$$

et Gudmundsson et al. [62] ont exploité ce résultat pour en dériver une autre. En notant $\Omega = \sum_{a\in A} \omega(a)$, $\hat{\Delta} = \max_{v\in V}\{\deg^-(v,\omega) + \deg^+(v,\omega)\}$ et $\hat{\omega} = \max_{a\in A}\{\omega(a)\}$, pour tout $\varepsilon > 0$, on a :

$$\chi_1(G,\omega) \leq \min\left\{ \left\lfloor 2\sqrt{2\Omega} + 1 \right\rfloor, \left\lceil \frac{\hat{\Delta}}{\left\lfloor \frac{1-\varepsilon}{\hat{\omega}} \right\rfloor + 1} \right\rceil \right\}.$$

Ces bornes sont d'autant plus intéressantes que le paramètre $\chi_\theta(G,\omega)$ est lié à $\alpha_k^\theta(G,\omega)$. On peut en effet aisément établir les relations suivantes avec les mêmes arguments qu'à la section 2.4.1 :

$$\begin{cases} \alpha_k^\theta(G,\omega) \leq n \leq \chi_\theta(G,\omega) \cdot \alpha_1^\theta(G,\omega) \\ \alpha_k^\theta(G,\omega) \leq k + n - \chi_\theta(G,\omega) \end{cases}$$

Nous verrons comment adapter simplement cette dernière définition du degré lorsque le graphe orienté possède également des poids sur les sommets. Cette généralisation sera notamment exploitée dans le Chapitre 4 pour développer des algorithmes qui se basent sur ce concept.

Bornes sur $\alpha_k(G)$

Revenons au cas non orienté. Les travaux de Narasimhan [53] sont consacrés à l'étude de $\alpha_k(G)$ sur des graphes classiques, on y trouve entre autres une borne supérieure sur ce paramètre. Il s'agit d'une généralisation du théorème du sandwich de Lovász [63], qui fait appel à une fonction $\vartheta(G)$ définie comme la plus grande valeur propre d'une matrice A parmi l'ensemble $\mathcal{A}(G)$ de toutes les matrices réelles symétriques telles que :
– $a_{ij} = 1$ si $\{i,j\} \in E$ ou si $i = j$;
– les autres coefficients sont quelconques, tant que la matrice reste réelle et symétrique.

De façon synthétique, la fonction de Lovász peut s'écrire de la façon suivante :

$$\vartheta(G) = \min_{A \in \mathcal{A}(G)} \left\{ \max_{\lambda} \{\lambda \mid \det(A - \lambda I) = 0\} \right\}.$$

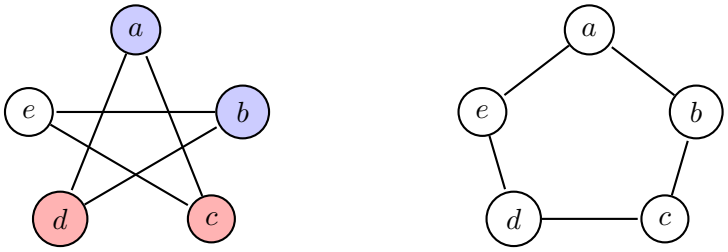
Parmi les nombreuses particularités de la fonction $\vartheta(G)$, le théorème du sandwich [63] de Lovász stipule que $\vartheta(G)$ est calculable en temps polynomial, tout en étant toujours comprise entre deux paramètres qui ne le sont pas : la taille du plus grand stable du complémentaire de G, et le nombre chromatique :

$$\alpha(\bar{G}) \leq \vartheta(G) \leq \chi(G).$$

La généralisation de ce théorème, proposée dans [53], permet d'obtenir une borne supérieure calculable en temps polynomial sur $\alpha_k(G)$:

$$\alpha_k(G) \leq \vartheta_k(\bar{G}),$$

où $\vartheta_k(\bar{G})$ minimise sur $\mathcal{A}(\bar{G})$ la somme des k plus grandes valeurs propres. Un exemple est illustré à la Figure 2.15 ci-dessous. On cherche le plus grand sous-graphe induit 2-colorable dans le graphe G, dont le complémentaire est le pentagone C_5. Il est possible de montrer [53] que $\vartheta_2(C_5) = 2\sqrt{5}$, et que donc $\alpha_2(G) \leq \left\lfloor 2\sqrt{5} \right\rfloor = 4$. Cette borne peut-être atteinte en colorant par exemple a et b d'une couleur, et c et d d'une autre.



$G(V, E)$ $\qquad$ $\bar{G}(V, E) = C_5$

Figure 2.15 Un exemple : $\alpha_2(G) \leq \left\lfloor \vartheta_2(\bar{G}) \right\rfloor = \left\lfloor 2\sqrt{5} \right\rfloor = 4$

Nous verrons au Chapitre 3 comment adapter cette borne basée sur la fonction généralisée de Lovász $\vartheta_k(G)$ pour les graphes orientés, complets, pondérés sur les arêtes et les sommets, dans le but d'approcher le paramètre $\alpha_k^{\theta}(G, W, \omega)$.

2.5.3 Structure de graphes

Classes de graphes

Parfois, il est possible d'exploiter la structure inhérente aux graphes étudiés pour améliorer ce type de bornes. En effet, certains problèmes de théorie des graphes sont $\mathcal{NP}$-difficiles dans le cas général, mais perdent ce caractère dès lors que l'on se restreint à certaines classes de graphes.

Cette stratégie a été utilisée dans [64] pour résoudre un problème d'affectation de fréquences dans un réseau sans fil. Les connexions en cours sont modélisées à l'aide d'un graphe d'intervalles, c'est-à-dire qu'il existe une bijection entre les sommets du graphe et un ensemble d'intervalles sur la droite réelle, de sorte que deux sommets sont adjacents si et seulement si l'intersection de leurs intervalles est non vide. Le fait de fixer la structure du graphe rend alors la coloration plus facile. En effet, il est possible de montrer [50] qu'un algorithme glouton sur un ordre particulier des sommets donne alors lieu à un coloration optimale.

Les graphes d'intervalles sont un cas particulier d'une famille particulièrement intéressante portant le nom de graphes parfaits.

Définition 17 (Graphe parfait)**.** *Un graphe est parfait si et seulement si ni lui ni son complémentaire ne contiennent de cycle impair induit de longueur au moins cinq.*

Lovász [63] a montré le résultat suivant.

Proposition 2. *Si G est un graphe parfait, alors*

$$\alpha(\bar{G}) = \vartheta(G) = \chi(G).$$

Autrement dit, si G est parfait, alors il est possible de déterminer les paramètres $\chi(G)$ et $\alpha(G)$ en temps polynomial (via le calcul de $\vartheta(G)$).

Dans cet esprit, Araujo et al. ont caractérisé le nombre chromatique θ-impropre sur des graphes appelés grilles (carrées, triangulaires et hexagonales), typiquement utilisés pour modéliser des réseaux d'antennes satellites.

Définition 18 (Carré d'un graphe)**.** *Soit $G = (V, E)$ un graphe pondéré par la fonction $\omega : E \to 1$. Le carré de G, noté G^2 est le graphe obtenu en ajoutant à G une arête de poids $1/2$ entre toute paire de sommets éloignés d'une distance égale à 2.*

Un exemple est illustré à la Figure 2.16.

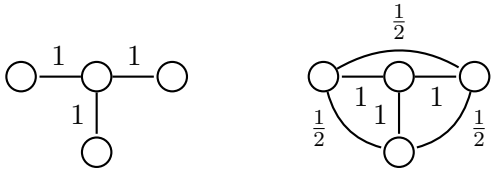


Figure 2.16 Un graphe G à gauche, et son carré G^2, à droite

Définition 19 (Grille carrée)**.** *Une grille carrée est le graphe dont tous les sommets sont des combinaisons linéaires entières des vecteurs* $(1,0)$ *et* $(0,1)$*. Deux sommets sont alors voisins si leur distance Euclidienne est égale à 1, ce qui donne 4 voisins pour un sommet* (a,b) *: les sommets* $(a-1,b)$*,* $(a+1,b)$*,* $(a,b-1)$ *et* $(a,b+1)$*.*

Proposition 3. *Soit G une grille carrée infinie. Alors*

$$\chi_\theta(G^2,\omega) = \begin{cases} 5 & si \quad 0 \leq \theta < \frac{1}{2}; \\ 4 & si \quad \frac{1}{2} \leq \theta < 1; \\ 3 & si \quad 1 \leq \theta < 3; \\ 2 & si \quad 3 \leq \theta < 8; \\ 1 & si \quad 8 \leq \theta; \end{cases}$$

La preuve, non triviale, se trouve dans [46]. On y trouve également le même type de caractérisations pour les grilles triangulaires et hexagonales. Cette approche est également privilégiée dans [62, 65], où $\chi_\theta(G,\omega)$ est déterminé sur des classes particulières de graphes.

Pavages de Voronoï

Les graphes utilisés ici sont issus d'usagers et de stations de base réparties aléatoirement dans le plan, donnant lieu à des pavages du plan qui vont s'avérer déterminants pour améliorer les heuristiques développées ici. Nous avons vu précédemment que chaque station de base définit un hexagone, qui correspond à l'ensemble des points les plus proches de la station. On peut généraliser ce concept dans le cas où les stations de base ne sont pas réparties de façon uniforme, et on obtient alors des cellules qui ne sont plus hexagonales, mais qui ont des formes quelconques (voir Figure 2.4). Une telle partition du plan est appelée *pavage de Voronoï* [66].

Définition 20 (Région de Voronoï)**.** *Soit S un ensemble fini de n points du plan. Une région de Voronoï associée à un élément p de S est l'ensembe des points qui sont plus proches de p que de tout autre point de S :*

$$Vor_S(p) = \left\{ x \in \mathbb{R}^2 |\ \forall q \in S\ d(x,p) \leq d(x,q) \right\}$$

où $d(x,y)$ représente la distance Euclidienne entre deux points du plan, x et y. Une partition du plan en régions de Voronoï est appelée pavage de Voronoï.

De nombreux algorithmes permettent d'obtenir ces diagrammes en $\mathcal{O}(n \log n)$ [66] qui s'avèrent extrêmement riches. Par exemple, il est alors facile de déterminer l'enveloppe convexe des n points, ou encore la plus courte distance entre toute paire de ces points, informations pouvant s'avérer coûteuses si l'on s'y prend différemment.

Ces pavages de Voronoï permettent donc de caractériser en partie les graphes sur lesquels on va chercher $\alpha_k^\theta(G, W, \omega)$, et nous verrons comment cela peut-être utile pour améliorer la précision des heuristiques développées dans le Chapitre 4.

2.5.4 Heuristiques de coloration constructives usuelles

Enfin, la dernière option pour résoudre un problème $\mathcal{NP}$-difficile consiste à utiliser des algorithmes qui, à défaut de pouvoir trouver systématiquement la solution optimale, trouvent des solutions de bonne qualité avec une probabilité élevée. En d'autres termes, il s'agit d'une autre méthode pour trouver une borne inférieure (supérieure) sur la fonction objectif s'il on traite un problème de maximisation (minimisation). Bien souvent ces algorithmes, ou *heuristiques*, compensent le fait qu'elles n'atteignent pas la solution optimale par leur rapidité.

Nous passons ici en revue les heuristiques constructives les plus connues, qui sont à la base des algorithmes présentés au Chapitre 4. Ce sont des algorithmes dits constructifs car ils construisent pas à pas une solution réalisable, développés en vue de minimiser le nombre de couleurs nécessaires pour colorer tous les sommets. Ils ont tous un caractère glouton : une couleur est assignée à chaque sommet de façon séquentielle, suivant un ordre déterminé par une certain règle de priorité. La nature de cette règle de priorité différencie les algorithmes entre eux et peut donner lieu à un ordre statique ou dynamique des sommets. La forme générique de ces algorithmes est résumée dans le Tableau 2.1 .

Tableau 2.1 Coloration séquentielle des sommets

Algorithme 1: Coloration séquentielle

```
1 INITIALISATION :
2 Déterminer un ordre des sommets à traiter à l'aide d'une certaine règle de priorité ;
3 COLORATION :
4 tant que tous les sommets ne sont pas colorés faire
5 |   Colorer le sommet prioritaire avec la plus petite couleur possible ;
6 |   Au besoin, mettre à jour les priorités ;
```

Voyons quelles sont les différentes règles de priorité envisageables. Premièrement, une série de quatre algorithmes dont l'ordre de traitement des sommets est statique, prédéterminé à l'avance, et un algorithme dont l'ordre est dynamique :

- *Random Sequential* (RS) : l'ordre est aléatoire ;
- *Largest First* (LF) : introduit par Welsh et Powell [67] en 1967, les sommets sont ordonnés par ordre de degré non croissant, l'idée étant de colorer en priorité les sommets ayant le plus de voisins ;
- *Smallest Last* (SL) : proposé par Matula et al. [68] en 1983, l'idée est similaire à LF, mais la mise en oeuvre est différente : l'ordre $v_1, v_2, \cdots, v_n$ des sommets est tel que v_i a le plus petit degré dans le graphe ne contenant plus que les sommets $v_1, v_2, \cdots, v_i$;
- *Connected Sequential* (CS) : à la différence de LF et SL qui se basent sur le nombre de voisins d'un sommet, cet algorithme, suggéré par Hertz et de Werra en 1989 [69], donne la priorité à un sommet dont les voisins sont déjà colorés, le privant ainsi de leur couleur : l'ordre $v_1, v_2, \cdots, v_n$ des sommets est tel que v_i a au moins un voisin dans la séquence $v_1, v_2, \cdots, v_{i-1}$;
- *Saturation Largest First* (SLF) : introduit par Brélaz [70] en 1979, cet algorithme exploite la même logique que pour CS, mais de façon légèrement plus sophistiquée : l'ordre de traitement des sommets est dynamique et change à chaque itération. Il se base sur la notion de degré de saturation d'un sommet, qui correspond au nombre de couleurs utilisées parmi ses sommets voisins. SLF donne alors priorité au sommet de plus grand degré de saturation, car il est privé d'un maximum de couleurs parmi celles déjà utilisées. Cet algorithme porte également le nom DSATUR, lorsque les cas d'égalité sont départagés par le sommet de plus grand degré (plutôt qu'aléatoirement). La notion de degré de saturation est généralisable, et de nombreux problèmes d'affectation de canaux ont exploité cette stratégie tel que mentionné dans le rapport [21].

Les méthodes précédentes considèrent donc une séquence des sommets, et attribuent la plus petite couleur possible au sommet traité. Une autre logique possible consiste à considérer séquentiellement les couleurs, et à construire successivement des ensembles stables, tel que résumé à la Table 2.2 ci-dessous.

Tableau 2.2 Construction séquentielle des classes de couleur

Algorithme 2: Construction séquentielle d'ensembles stables

```
1 INITIALISATION :
2 Déterminer un ordre des sommets à traiter à l'aide d'une certaine règle de priorité ;
3 i = 1 ;
4 COLORATION :
5 tant que tous les sommets ne sont pas colorés faire
6 |   C_i = ∅ ;
7 |   Saturer la classe C_i en considérant les sommets par priorité non croissante ;
8 |   Au besoin, mettre à jour les priorités ;
9 |   i = i + 1 ;
```

Deux algorithmes principaux sont basés sur cette stratégie :

– *Greedy Independent Sets* (GIS) mis au point par Johnson [71] en 1974, cette procédure consiste à construire successivement des stables maximums par l'inclusion. Le premier sommet à être introduit dans une classe est choisi parmi ceux ayant un nombre maximum de voisins colorés, et les prochains sommets rajoutés à la classe sont ceux ayant un nombre minimum de voisins candidats pour cette classe. L'idée est de priver la couleur utilisée pour un minimum de sommets lorsque la classe est en cours de construction, de façon à avoir des ensembles stables les plus grands possibles ;

– *Recursive Largest First* (RLF) : développé par Leighton [72] cinq ans plus tard (la même année que SLF), cet algorithme a la même structure que GIS, mais effectue ses choix différement. Contrairement à GIS, le premier sommet à être introduit dans une classe est choisi parmi ceux ayant un nombre maximum de voisins non colorés, et les prochains sommets rajoutés à la classe sont ceux ayant un nombre maximum de voisins non colorés qui ne peuvent plus faire partie de cette classe.

Il n'est pas difficile d'adapter ces stratégies pour la coloration θ-impropre sur des graphes pondérés. Araujo et al. [46] ont développé une heuristique dite de *nivellement* pour minimiser $\theta \in \mathbb{R}^+$. Voici son principe : les sommets prioritaires sont les sommets v qui maximisent la

quantité

$$I(v) = \sum_{i=1}^{k} \sum_{u \in N_v | c(u)=i} \omega(u, v),$$

autrement dit ceux de plus grand degré pondéré par rapport aux sommets déjà colorés, et les couleurs prioritaires sont les couleurs i qui minimisent l'interférence résultante sur le sommet traité, donnée par la quantité

$$I_i(v) = \sum_{u \in N_v | c(u)=i} \omega(u, v).$$

Bien que cette heuristique soit différente des précédentes, dans la mesure où les graphes sont pondérés et que le paramètre θ entre en jeu, l'idée générale reste la même : les sommets sont traités séquentiellement par une règle de priorité, et une couleur est attribuée suivant un autre critère.

Des stratégies similaires, basées sur les algorithmes LF, SLF et RLF, sont proposées dans le Chapitre 4 sur des graphes complets, orientés et pondérés.

Tous ces algorithmes étant semblables mais néanmois distincts, il est légitime de se demander lequel est le plus performant. Ceci fait l'objet de la sous-section qui suit.

2.6 Comment mesurer la performance d'un algorithme ?

Un algorithme est généralement évalué suivant deux critères : sa rapidité et sa précision. Mesurer la rapidité d'un algorithme n'est pas chose aisée, notamment car cela peut dépendre de la façon dont celui-ci a été implanté, du langage de programmation utilisé, de la puissance de calcul de l'ordinateur, etc. Pour remédier à cela, on s'intéresse avant tout à sa complexité algorithmique (voir Définition 9), qui fournit indirectement un ordre de grandeur sur le temps d'exécution théorique. Quant à la précision, elle est évaluée à deux échelles différentes : lorsque le nombre de sommets n tend vers l'infini, via la notion de *garantie*, et lorsque n tend vers 0, via celle de *plus petit graphe sous-optimal*.

2.6.1 Complexité

Le tableau 2.3 ci-après, basé sur le rapport [73] résume les complexités algorithmiques des heuristiques introduites précédemment.

Tableau 2.3 Complexités algorithmiques des heuristiques de coloration usuelles

Algorithme	RS, LF, SL, CS	SLF	GIS, RLF
Complexité	$\mathcal{O}(m+n)$	$\mathcal{O}\left((m+n)\log n\right)$	$\mathcal{O}(mn)$

Ce sont donc des heuristiques très rapides. Pour avoir une idée ce que cela représente en temps de calculs, d'après les tests numériques récents de [74], des graphes à 1 000 sommets de densité 0,9 [5] sont colorés avec l'algorithme RLF en environ une seconde. Rappelons qu'il est à ce jour impossible de colorer des graphes de cette densité à plus de 100 sommets de façon exacte.

2.6.2 Garantie

Pour mesurer la précision d'une méthode heuristique, on s'intéresse à sa *garantie*, qui est essentiellement le ratio entre la valeur de la solution fournie par l'heuristique, et celle de la solution optimale.

Définition 21 (Garantie)**.** *Soit une instance I d'un problème de minimisation parmi l'ensemble $\mathcal{I}$ de toutes les instances possibles. Soit OPT(I) la valeur de la solution optimale et A(I) celle fournie par un algorithme A. Alors A garantit $f(I)$ si $f(I)$ est une fonction à valeurs dans $[1,\infty[$ telle que*

$$\frac{A(I)}{OPT(I)} \leq f(I) \quad \forall I \in \mathcal{I}.$$

Généralement, ces rapports sont étudiés lorsque n tend vers l'infini, et l'on s'intéresse alors à l'ordre de grandeur obtenu. De plus, le ratio de performance absolue de A est défini par $r_A := \max_{I\in\mathcal{I}}\left\{\frac{A(I)}{OPT(I)}\right\}$, et si $r_A = \varepsilon$, on dit que A est un algorithme ε-compétitif.

Garanties pour $\chi(G)$

La meilleure garantie connue à ce jour pour le nombre chromatique (par un algorithme polynomial) est de l'ordre de $\mathcal{O}\left(\frac{n(\log\log n)^2}{\log^3 n}\right)$, et il existe une conjecture [75] sur la meilleure garantie possible : $\mathcal{O}\left(\frac{n}{\log^c n}\right)$, où c est une constante supérieure ou égale à 3. Il est en fait très curieux que la meilleure garantie possible est obtenue au moyen d'un dérivé de l'algorithme GIS de Johnson, noté GIS'. En effet, il est possible de montrer [73] que celui-ci a une garantie en $\mathcal{O}\left(\frac{n}{\log n}\right)$, ratio qui a été amélioré par Wigderson [76] à $\mathcal{O}\left(\frac{n(\log\log n)^2}{\log^2 n}\right)$ puis par Berger et Rompel. [77] à $\mathcal{O}\left(\frac{n(\log\log n)^3}{\log^3 n}\right)$.

5. La densité est la probabilité qu'une paire quelconque de sommets soit reliée par une arête.

Nous avons vu que les algorithmes constructifs RS, LF, SL, CS, colorient séquentiellement les sommets, et que donc un maximum de $\Delta(G)$ couleurs peuvent déjà apparaître dans le voisinage d'un sommet donné, donnant lieu à des colorations avec au plus $\Delta(G)+1$ couleurs. Mais comme $\Delta(G)$ peut être arbitrairement grand (par exemple pour une étoile à m branches), la coloration obtenue peut être de très mauvaise qualité si cette borne est atteinte. Il est en fait possible de montrer [73] que ces algorithmes ont une garantie en $\mathcal{O}(n)$, ce qui correspond à la pire garantie possible pour une coloration d'un graphe à $\mathcal{O}(n)$ sommets. On pourrait croire que les algorithmes SLF et RLF (qui ont une moins bonne complexité) font mieux, mais il n'en est rien : leur garantie est également en $\mathcal{O}(n)$. Seul GIS assure de colorer un graphe en au plus $\mathcal{O}\left(\frac{n}{\log n}\right)\chi(G)$ couleurs. Ces résultats sont regroupés dans le Tableau 2.4 ci-dessous.

Tableau 2.4 Garanties des heuristiques de coloration usuelles

Algorithme	RS, LF, SL, CS, SLF, RLF	GIS	GIS'
Garantie	$\mathcal{O}(n)$	$\mathcal{O}\left(\frac{n}{\log n}\right)$	$\mathcal{O}\left(\frac{n(\log\log n)^2}{\log^3 n}\right)$

Ces garanties sont donc mauvaises dans le cas général, mais en se restreignant à certaines classes de graphes, ces heuristiques peuvent être optimales. Par exemple, l'algorithme SLF est optimal pour les graphes bipartis, les arbres, les cycles, les colliers, les cactus [73].

Garanties pour $\alpha_k(G)$

Nous avons vu que la méthode GIS a une garantie meilleure que les autres heuristiques usuelles pour approcher le paramètre $\chi(G)$. Si l'on cherche le plus grand sous-graphe induit k-colorable, il est légitime de se demander si cette stratégie est intéressante. Dans cet esprit, Narasimhan [53] répond à la question suivante : supposons que l'on dispose d'un algorithme exact pour trouver le stable maximum ; est-il judicieux de construire séquentiellement les k stables maximums pour approcher $\alpha_k(G)$? Il se trouve qu'une telle stratégie garantit [6] un ratio de performance égal à $\frac{k+1}{2k}$, ce qui signifie que la valeur optimale $\alpha_k(G)$ est dans le pire des cas $\frac{2k}{k+1}$ fois plus grande que la valeur obtenue en construisant séquentiellement des stables maximums. Narasimhan propose une famille de graphes critiques où cette borne est

6. Pour un problème de maximisation, on dit qu'un algorithme A garantit $f(I)$ si $f(I)$ est une fonction à valeurs dans $[0,1]$ telle que

$$\frac{A(I)}{OPT(I)} \geq f(I) \quad \forall I \in \mathcal{I}.$$

atteinte asymptotiquement. La Figure 2.17 en illustre un. Tel qu'illustré à gauche, le graphe est entièrement 2-colorable , mais l'algorithme glouton va renvoyer par exemple l'ensemble $\{\{a, e, c, f\}, \{b\}\}$, de taille 5 (à droite). En remplaçant les stables $\{a, e\}$ et $\{c, f\}$ par des ensembles de $n + 1$ sommets indépendants, et les sommets $\{b\}$, $\{d\}$ par des ensembles de n sommets indépendants, on obtient effectivement un ratio de $\frac{4n+2}{3n+2}$, qui tend bien vers $\frac{4}{3}$, soit $\frac{2k}{k+1}$ avec $k = 2$.

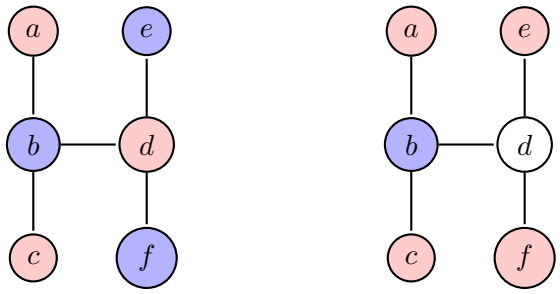


Figure 2.17 Un exemple : $\frac{A}{\alpha_2(G)} = \frac{5}{6}$.

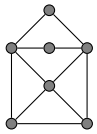
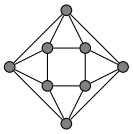
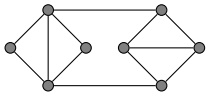
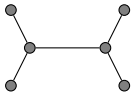
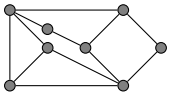
2.6.3 Plus petit graphe sous-optimal

Les garanties de performance sont de nature asymptotique, puisque l'on compare les ratios lorsque n tend vers l'infini. Ceci présente l'inconvénient de ne pas être représentatif des cas où n est très petit, à l'autre extrémité de l'échelle. De nombreuses études empiriques telles que celle tirée de [78] ont d'ailleurs montré que l'algorithme RLF est généralement plus performant que LF, SL, RS, LSF, RS, bien que ces méthodes aient toute la même garantie.

Hansen et Kuplinsky [79] ont partiellement remédié à cela en introduisant la notion de plus petit *graphe sous-optimal* d'un algorithme. Un graphe est défini comme tel lorsque toute implémentation d'un algorithme fournit une solution de taille strictement supérieure à $\chi(G)$. On s'intéresse alors au plus petit graphe sous-optimal, autrement dit celui avec le moins de sommets n et d'arêtes m possibles (on minimise d'abord n, puis m). L'analyse de ces graphes peut ainsi être vue comme un outil complémentaire à celui de garantie de performance.

Le Tableau 2.5 ci-dessous, tiré de [80], présente ces graphes pour les heuristiques de coloration usuelles. Il est curieux que le plus petit graphe sous-optimal de CS soit encore inconnu ; Babel et al. [81] ont présenté un graphe cubique à 18 sommets et ont prouvé qu'il s'agit du plus petit graphe sous-optimal de CS pour cette classe de graphes, et ont émis la conjecture qu'il s'agit du plus petit graphe sous-optimal dans le cas général. À ce jour, la question demeure ouverte. Notons également qu'un tel graphe ne peut exister lorsque l'ordre des sommets est choisi aléatoirement.

Tableau 2.5 Plus petits graphes sous-optimaux des heuristiques de coloration usuelles

Algorithme	RS	LF	SL	CS	SLF	GIS	RLF
G	∅			?			

2.6.4 Performance numérique

Ces trois critères - complexité, garantie et plus petit graphe sous-optimal - sont des outils de mesure de performance théorique. Tel que mentionné plus haut, l'algorithme GIS a la meilleure garantie de performance, mais les tests numériques semblent montrer que c'est RLF qui en moyenne, se rapproche le plus souvent de la solution optimale. De même, la complexité algorithmique évalue un ordre de grandeur sur le nombre d'opérations élémentaires requises dans le pire des cas, et n'est pas toujours fidèle à la réalité. Par exemple, l'algorithme du simplexe a une complexité exponentielle, mais en pratique il se comporte très bien et s'avère extrêmement efficace pour des problèmes linéaires de grande taille. Il est donc également indispensable d'évaluer un algorithme numériquement, si possible sur des instances publiques telles que les instances DIMACS [7] pour la coloration de sommets.

Pour chacune des heuristiques proposées dans le Chapitre 4 pour approcher le paramètre $\alpha_k^\theta(G, W, \omega)$, nous verrons quelle est leur complexité, leur garantie, ainsi que leur plus petit graphe sous-optimal. Nous mettrons également en évidence comment les améliorer en exploitant la structure des réseaux de téléphonie mobile, notamment à l'aide de pavages de Voronoï. Des simulations numériques viendront compléter l'analyse de ces algorithmes.

2.7 Coloration et métaheuristiques

Dans la section précédente, nous avons passé en revue les heuristiques constructives usuelles pour la coloration de graphes. Il existe une autre grande famille d'algorithmes souvent utilisée pour résoudre des problèmes $\mathcal{NP}$-difficiles : les métaheuristiques.

Il s'agit de stratégies génériques, c'est-à-dire pouvant a priori s'appliquer à n'importe quel problème combinatoire (y compris les problèmes d'affectation de canaux, voir [21]), qui permettent de guider la recherche d'une solution optimale. Leur spécificité est d'explorer l'espace

7. `http://mat.gsia.cmu.edu/COLOR/instances.html`

de recherche (qui, on le rappelle, est souvent trop grand pour être parcouru entièrement) de façon efficace, dans le but de trouver une solution de bonne qualité. Il n'existe aucune garantie de convergence ou d'optimalité. Les techniques qui constituent cette famille d'algorithmes sont très variées et constituent une branche des mathématiques discrètes à part entière. Nous allons nous concentrer sur l'une d'entre-elles, appelée Recherche Tabou, qui est une des plus populaires pour la coloration de graphes.

Recherche Tabou

Développée par Glover [82], la Recherche Tabou est une métaheuristique basée sur une idée relativement simple. À partir d'une solution initiale quelconque, l'algorithme explore le voisinage de cette solution, jusqu'à trouver un optimum local : on parle de descente. Par exemple, supposons que l'on cherche le nombre chromatique d'un graphe. À partir d'une coloration quelconque (sans conflits), une descente pourrait consister à considérer chaque couleur, et à essayer de l'éliminer en parcourant chacun des sommets affectés de cette couleur en essayant de la remplacer par une autre. Si une couleur est éliminée, la coloration correspondante est dite voisine de la solution initiale. Si aucune couleur ne peut être éliminée, on est sur un minimum local. Ceci constitue une itération de la Recherche Tabou. Pour éviter de rester piégé dans ce minimum local, la Recherche Tabou exige à l'itération suivante de faire une autre descente vers la solution voisine qui dégrade le moins la solution courante. Ceci signifie que l'on se déplace toujours d'une coloration vers une coloration voisine, quitte à ce que cela n'améliore aucunement la solution. Ce mécanisme permet de ne pas rester piégé dans un minimum local. Lorsque l'on quitte un tel minimum, il faut alors éviter d'y retourner. Pour cela, la Recherche Tabou donne alors le statut *tabou* à la solution correspondant à cet optimum local, et y interdit tout déplacement pendant un nombre fixé d'itérations. Tel est le fonctionnement général de cette métaheuristique, qui de façon itérative, parcourt l'espace des solutions et garde en mémoire la meilleure rencontrée. Un critère d'arrêt arbitraire (un certain temps écoulé, un nombre limite d'itérations, etc.) permet de mettre fin à l'exploration.

Recherche Tabou et $\chi(G)$

Tel qu'évoque plus haut, cette stratégie peut s'appliquer à n'importe quel problème de nature combinatoire, mais son efficacité dépend bien souvent de la façon dont sont définis l'espace de recherche, les voisinages et la fonction objectif du problème. Par exemple, l'exemple décrit précédemment n'est en réalité pas du tout efficace pour minimiser le nombre de couleurs utilisées. À ce jour, une des meilleures mises en oeuvre de recherche tabou pour la recherche du nombre chromatique s'appelle TABUCOL et a été développé par Hertz et de Werra [83].

La particularité de TABUCOL est de fixer le nombre de couleurs à une borne supérieure sur le nombre chromatique (par exemple à $\Delta + 1$), et de partir d'une solution initiale qui n'est pas nécessairement réalisable. Une solution voisine est obtenue en attribuant à un sommet conflictuel une autre couleur. La fonction objectif est alors le nombre de conflits, et la métaheuristique vise à minimiser le nombre de conflits jusqu'à trouver une solution réalisable. Le cas échéant, une couleur est éliminée, et la procédure cherche à nouveau une solution réalisable avec une couleur en moins, et ainsi de suite.

Recherche Tabou et $\alpha(G)$

Friden, Hertz et de Werra [84] ont également développé une Recherche Tabou pour le problème du stable maximum, portant le nom de STABULUS. Bien que similaire à celle de TABUCOL, sa structure mérite d'être décrite. Cette fois-ci, la coloration n'utilise qu'une seule couleur, et le paramètre que l'on incrémente itérativement est la taille du stable. La solution initiale est donc un ensemble de sommets colorés, et la fonction objectif est à nouveau le nombre de conflits. La structure de voisinage est définie de la façon suivante : une solution voisine est obtenue en permuttant un sommet coloré et un sommet non coloré. Lorsqu'une solution réalisable est trouvée, un sommet supplémentaire est ajoutée au stable courant, et ainsi de suite.

Dans le Chapitre 5, nous ferons usage d'une variante de TABUCOL pour colorer les sommets qui apparaissent au fur et à mesure dans le graphe dans le cadre d'une coloration dite on-line, introduite à la section suivante.

2.8 Coloration de graphes avec données incomplètes

Jusqu'à présent, nous avons considéré que les graphes étudiés étaient entièrement connus avant l'exécution de l'algorithme de coloration utilisé. Or, il peut arriver que certaines informations apparaissent au fur et à mesure, que celles-ci changent au cours du temps ou encore que l'information ne soit pas disponible à tous les sommets du graphe.

2.8.1 Coloration on-line

Les problèmes dits *on-line* sont caractérisés par le fait que les données du problème (les sommets ou les arêtes par exemple) ne sont pas entièrement connues à l'avance, mais sont révélées étape par étape sur un horizon roulant. Le but est alors de construire la meilleure

solution possible au fur et à mesure. Dans le cas classique, les décisions à chaque étape sont définitives. Autrement dit, un algorithme de coloration on-line reçoit les sommets dans un ordre $v_1, \cdots, v_n$, et la couleur attribuée au sommet v_i ne peut se faire que sur la base du sous-graphe induit par les sommets $\{v_1, \cdots, v_i\}$. Il est clair que ceci est plus difficile, dans la mesure où l'algorithme utilisé est en quelque sorte aveugle et a donc plus de chances d'effectuer une mauvaise décision. L'optimisation de l'affectation de ressources dans un réseau de téléphonie mobile rentre justement dans ce cadre, puisque les usagers (les sommets du graphe) peuvent apparaître à des instants différents dans le réseau.

Méthodologie pour l'algorithmique on-line

Les problèmes d'optimisaton on-line peuvent être vus comme un jeu opposant deux adversaires : l'algorithme on-line, qui essaye de construire la meilleure solution possible avec les informations dont il dispose, et le *spoiler*, qui révèle des informations à chaque itération de façon à piéger son adversaire. Cette approche est privilégiée pour construire toutes les preuves des ratios de performances, nous verrons quelques exemples ci-après.

Algorithmes on-line et $\chi(G)$

Il existe quelques familles de graphes pour lesquelles il existe des algorithmes on-line compétitifs pour la recherche du nombre chromatique. Par exemple, si le graphe G est biparti (i.e., s'il est colorable en deux couleurs), alors il existe un algorithme qui garantit de trouver une coloration avec au plus $n \log 2$ couleurs. Les familles de graphes suivantes en possèdent : les graphes sans P_4 [85], les graphes sans P_5 [85, 86], les graphes sans $2K_2$ [85], les complémentaires de graphes bipartis et triangulés [85], les graphes d'intervalles [87], les graphes d'intervalles circulaires [88]. En revanche, de simples extensions à ses familles telles que les arbres ou les graphes sans P_6 n'en possèdent pas [85]. De manière générale, il n'existe pas d'algorithme on-line compétitif pour un graphe quelconque, alors que tout graphe quelconque est $(\Delta+1)$-colorable de façon off-line (nous avons vu que n'importe quel algorithme séquentiel de coloration y parvient). La coloration on-line est bel et bien plus difficile que la coloration usuelle.

Pour ce type de problématique, une première idée intuitive consiste à utiliser des algorithmes gloutons, qui traitent les sommets séquentiellement, en leur attribuant une couleur en fonction d'une certaine règle de priorité. Différentes règles de priorité ont été envisagées dans [89] :

– la plus petite couleur disponible est attribuée ; intuitivement, cela permet de saturer les classes successivement ;

– la couleur disponible dont la classe est la plus petite est attribuée ; inversement, ceci permet d'équilibrer la taille des classes au cours de la coloration ;
– la couleur disponible dont la classe est la plus grande est attribuée ; ceci permet aussi de saturer les classes le plus possible ;
– une couleur est attribuée aléatoirement parmi toutes celles disponibles ; si l'on reprend l'analogie avec un jeu à deux adversaires, cette stratégie rend la tâche difficile pour le spoiler ;

et si aucune couleur ne convient, une nouvelle couleur est utilisée. Les tests numériques de Ouerfelli et al. [89] laissent penser que les première et troisième stratégie sont les plus efficaces pour approcher le nombre chromatique.

Dans le Chapitre 5, d'autres stratégies de ce type seront envisagées pour la recherche du paramètre $\alpha_k(G)$ de façon on-line.

Algorithmes on-line et $\alpha_k(G)$

Il n'existe à notre connaissance pas d'algorithme on-line pour la recherche du plus grand sous-graphe induit k-colorable. Les travaux qui s'en rapprochent le plus pourraient être [90, 91], où le problème du stable maximum on-line est étudié en détail. Des algorithmes ainsi que leurs ratios de performance sont proposés pour différentes configurations on-line : les sommets ou les arêtes arrivant ou disparaissant par paquets. Rappelons quelques résultats importants de ces travaux, qui sont particulièrement intéressants, dans la mesure où ils fournissent des résultats théoriques ainsi que des algorithmes compétitifs pour un cas particulier du problème du plus grand sous-graphe induit k-colorable.

Supposons que les sommets arrivent un par un, il y a donc autant d'itérations que de sommets dans le graphe considéré. Lorsqu'un sommet apparaît, ses arêtes incidentes sont également révélées.

Proposition 4. *Aucun algorithme on-line ne peut garantir une solution de taille strictement supérieure à $\frac{\alpha(G)}{n-1}$ pour le problème du stable maximum.*

Démonstration. Quelque soient les choix de l'algorithme A on-line utilisé, il est possible de générer un graphe pour lequel $\alpha(G) = n - 1$, mais pour lequel l'algorithme A ne trouve pas de solution de taille supérieure à 1. Voici comment construire un tel graphe. Tant que l'algorithme A ne choisit pas de sommet, tous les sommets révélés $v_1, \cdot, v_i$ sont des sommets isolés. Dès que l'un d'entre eux est choisi, alors les sommets suivants $v_{i+1}, \cdots, v_n$ sont tous

reliés au sommet v_i, formant une étoile à $n-i$ branches. Au mieux, l'algorithme A ne peut donc sélectionner qu'un seul sommet (v_i), alors que le plus grand stable est clairement constitué de l'ensemble $\{v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n\}$. Un exemple est illustré à la Figure 2.18 ci-dessous avec $n = 5$. Le sommet v_2 est l'unique sommet choisi, alors que le stable maximum est constitué de l'ensemble $\{v_1, v_3, v_4, v_5\}$.

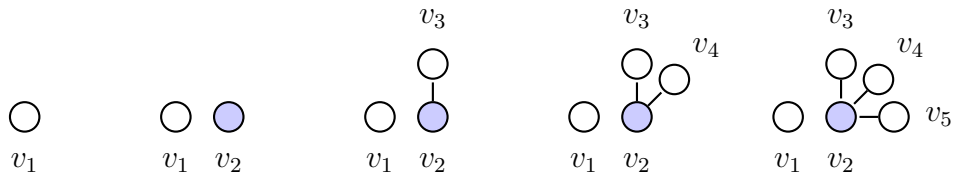


Figure 2.18 $\dfrac{A(G)}{\alpha(G)} = \dfrac{1}{5} < \dfrac{1}{4}$.

□

On peut noter que l'algorithme simpliste qui consiste à choisir le premier sommet apparu atteint trivialement cette garantie, de même que l'algorithme glouton qui incorpore un sommet révélé dans le stable courant si possible. En effet, étant donné un stable maximal pour l'inclusion $S = \{v_1, \cdots, v_{A(G)}\}$ et un stable maximum $\{u_1, \cdots, u_{\alpha(G)}\}$, soit u_i correspond à un des sommets de S, soit c'est un voisin d'au moins un sommet de S. Il s'ensuit donc que

$$\alpha(G) \leq \sum_{i=1}^{A(G)} \max\{1, \deg(v_i)\} \leq A(G)\Delta(G),$$

d'où le fait que $\frac{A(G)}{\alpha(G)} \geq \frac{1}{\Delta(G)}$.

Escoffier et al. [90, 91] traitent également le cas où à chaque itération, plusieurs sommets (ainsi que leurs arêtes incidentes) sont révélés. Après $t < n$ itérations, le graphe est entièrement connu. On a le résultat suivant.

Proposition 5. *Aucun algorithme on-line ne peut garantir une solution de taille strictement supérieure à $\frac{\alpha(G)}{\sqrt{n}-1}$ pour le problème du stable maximum.*

Démonstration. La proposition est déjà vraie pour le cas où $t = 2$. Voici comment construire le graphe en question. À la première itération un ensemble de $\lfloor\sqrt{n}\rfloor$ sommets isolés sont révélés. Deux cas de figures se présentent. Si au moins un sommet est choisi, alors il suffit de révéler le restant des sommets de sorte qu'ils soient isolés entre eux, mais tous reliés à chacun des sommets de la première itération. L'ensemble stable est ainsi saturé, alors que le deuxième ensemble révélé, de taille $n - \lfloor\sqrt{n}\rfloor$, est un ensemble stable de taille plus grande. On a bien

$$\frac{A(G)}{\alpha(G)} \leq \frac{\lfloor\sqrt{n}\rfloor}{n - \lfloor\sqrt{n}\rfloor} \leq \frac{1}{\sqrt{n}-1}.$$

Sinon, l'ensemble révélé est un sous-graphe complet, non connecté au premier ensemble. Dans ce cas, le stable ne peut comprendre qu'un seul sommet, et on a

$$\frac{A(G)}{\alpha(G)} \leq \frac{1}{\lfloor\sqrt{n}\rfloor} \leq \frac{1}{\sqrt{n}-1}.$$

Les trois configurations possibles sont illustrées à la Figure 2.19 ci-dessous avec $n = 5$.

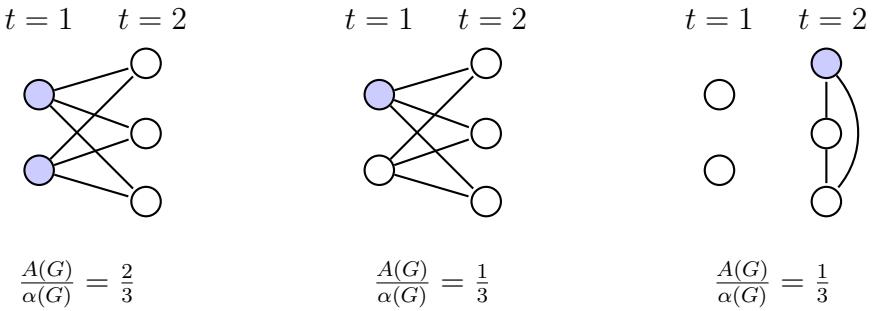


Figure 2.19 $\frac{A(G)}{\alpha(G)} < \frac{1}{\sqrt{5}-1}$

□

Ils prouvent également que cette garantie est atteinte asymptotiquement par l'algorithme qui à chaque itération trouve le stable maximum parmi les sommets révélés, et le renvoie si celui-ci est de taille supérieure ou égale à $\sqrt{\frac{n}{t}}$.

Ces résultats portant sur les garanties de performance d'un algorithme on-line pour la recherche de $\alpha(G)$ seront généralisés dans le Chapitre 5, lorsque l'on dispose de k couleurs.

Coloration on-line des arêtes

Dans cette thèse, nous nous concentrons sur la coloration des sommets, mais il est également possible de considérer la coloration des arêtes : deux arêtes ayant une extremité en commun doivent avoir des couleurs différentes. Favrholdt et al. [92] se sont intéressés à un problème étroitement lié au nôtre : celui de maximiser le nombre d'arêtes colorées étant donné un nombre de couleurs fixé k, et ceci dans une configuration on-line. Le lien entre ce problème et le nôtre motive la définition suivante.

Définition 22. *(Graphe de ligne) Soit un graphe G. Son graphe de ligne $L(G)$ est obtenu en associant un sommet à chaque arête de G et en reliant deux sommets dans $L(G)$ si et seulement si les arêtes correspondantes dans G ont une extrémité en commun.*

Il s'ensuit que la coloration des arêtes d'un graphe G est équivalente à la coloration des sommets de $L(G)$. Par exemple, colorer les arêtes d'un chemin à m arêtes est équivalent à colorer les sommets d'un chemin à m sommets (voir Figure 2.20). Favrholdt et al. [92] ont montré qu'avec $k = 2$ couleurs, aucun algorithme on-line ne peut garantir un ratio supérieur à $\frac{4}{5}$ pour ce cas de figure, et ont développé un algorithme aléatoire qui atteint cette garantie.

Figure 2.20 Une 2-coloration des arêtes d'un chemin G à gauche, et la coloration correspondante des sommets de son graphe de ligne $L(G)$ à droite

Relaxations

Dans la coloration on-line classique, les décisions prises à chaque étape sont irrévocables. Par conséquent un sommet non coloré l'est définitivement. Il est possible de relâcher cette contrainte, et de permettre de revenir sur une décision moyennant une pénalité. Cet aspect a également été abordé par Escoffier et al. [90, 91] pour le problème du stable maximum. Trois modèles ont été considérés :

– les sommets choisis tardivement par rapport au temps $\tau = 0$ sont pénalisés. Plus précisément, la valeur d'un sommet vaut 1 à la première itération, puis est divisée par une constante $p > 1$ à chaque itération ; à l'itération i, les sommets ont donc une valeur égale à p^{1-i}. Ils proposent un algorithme de garantie égale à

$$\max\left\{p^{1-t}, \max_{i=1,\cdots,t-1}\left\{\frac{p^{1-i}}{n-i+1}\right\}\right\},$$

et démontrent qu'on ne peut pas faire mieux ;

– les sommets choisis tardivement par rapport à leur date d'apparition sont pénalisés ; à la différence du modèle précédent, c'est le fait d'attendre l'incorporation d'un sommet dans la solution qui doit est pénalisé (ceci semble plus juste, dans la mesure où l'apparition tardive d'un sommet que l'on ne pouvait choisir plus tôt est indépendante de l'algorithme). Un sommet qui apparaît à l'itération j a donc une valeur égale à p^{j-i} à l'itération courante i. Ils proposent un algorithme de garantie égale à

$$\max\left\{p^{1-t}, \frac{1}{\sqrt{tn}}\right\},$$

et démontrent à nouveau qu'on ne peut faire mieux ;

– les sommets de l'itération précédente peuvent être supprimés du stable, sans pénalité. L'algorithme a face à lui à l'itération i un graphe dont les sommets révélés à l'itération j sont valués par p^{j-i} . Il peut alors choisir des sommets de ce graphe, ainsi qu'en enlever de l'itération précédente. L'algorithme construit ainsi sa solution, en prenant et en enlevant des sommets au fur et à mesure de la révélation. Ils proposent un algorithme de garantie égale à

$$\frac{1}{t - \frac{t-1}{p}},$$

et prouvent encore qu'il s'agit de la meilleure garantie possible.

2.8.2 Coloration dynamique

Dans la coloration on-line, on suppose que les données apparaissent au fur et à mesure. Il existe également des situations où les données peuvent disparaître entre deux itérations, et où les contraintes d'incompatibilité évoluent sur un horizon de temps. On parle alors de coloration dynamique. Un tel problème peut être défini par une séquence de graphes $G_t = (V(t), E(t))$, où t appartient à un horizon de temps T, chaque graphe G_t possédant sa propre topologie, et donc sa propre coloration optimale. À une itération donnée, la coloration doit être produite avant que le graphe suivant soit considéré. Tel que mentionné dans [93], la coloration dynamique permet de modéliser de nombreuses applications aussi bien dans le domaine de la biologie que de la chimie, des neuro-sciences, ou encore de la réseautique.

Algorithmes de coloration dynamique

Ce problème est abordé dans [94] dans le cas particulier où le nombre de sommets reste constant, mais les arêtes peuvent apparaître ou disparaître à chaque itération. L'algorithme proposé, baptisé ACODYGRA, colore les sommets à l'aide de l'algorithme RLF lors de la première itération. Pour les suivantes, un *agent* est placé sur chaque sommet et peut recolorer ce sommet ou ses voisins. Un sommet peut être recoloré pour deux raisons : soit une nouvelle arête a créé un conflit, soit la disparition d'une arête diminue le degré de saturation d'un sommet, et il est alors intéressant de reconsidérer une coloration en vue de diminuer le nombre total de couleurs utilisées. Les tests numériques montrent que si le nombre d'arêtes mises à jour est faible, cette approche est intéressante, dans la mesure où elle produit de bonnes solutions en un temps inférieur aux heuristiques de colorations usuelles (exécutées à chaque itération). Par contre, dès lors qu'un nombre important de mises à jour ont lieu, l'algorithme perd en compétitivité : le temps de résolution augmente et donne des colorations largement sous-optimales.

Le niveau de similarité entre deux graphes successifs de la séquence, G_i et G_{i+1} est ainsi une problématique inhérente à la coloration dynamique : s'ils sont similaires ou que les changements d'une itération à l'autre sont coûteux, il peut être intéressant d'utiliser la coloration de G_i comme solution initiale pour celle de G_{i+1}. S'ils sont très différents, ou que les coûts liés aux changements sont négligeables, il peut être préférable de considérer chaque graphe indépendamment l'un de l'autre. Cette question a été abordée notamment dans [95] pour traiter un problème d'affectation de fréquences dans une base militaire. La stratégie utilisée est la suivante : chaque nouvelle requête est traitée de façon gloutonne, avec différents critères :

- la plus petite couleur disponible attribuée ;
- la couleur permettant de maintenir un niveau maximal de *disponibilité* est sélectionnée. Ce critère, analogue au degré de saturation d'un sommet, est propre à la topologie du problème considéré et correspond à une mesure sur la capacité à recevoir de nouvelles connexions ;

et si l'algorithme ne parvient pas à satisfaire la nouvelle requête, un algorithme de réparation est exécuté au cours duquel les affectations précédentes sont réévaluées de façon à pouvoir intégrer cette requête dans le réseau. Le but est de trouver une solution réalisable en minimisant le nombre de changements ; à nouveau deux stratégies sont envisagées :

- par recherche locale, à l'aide d'un algorithme tabou ;
- par une méthode exacte, au moyen d'un algorithme de branchement avec une limite de temps.

Les tests numériques effectués montrent que le deuxième algorithme glouton (basé sur le critère de disponibilité) est plus efficace. Concernant la méthode de réparation, de façon prévisible, l'algorithme exact est plus précis mais plus coûteux en temps de calcul. Le meilleur compromis semble être d'utiliser un algorithme hybride, c'est-à-dire une méthode exacte assistée par une méthode de recherche locale tel qu'une Recherche Tabou.

Cette problématique sera abordée dans le Chapitre 5. En particulier, nous verrons comment adapter les heuristiques LF, SLF et RLF dans une configuration dynamique, et comment minimiser le nombre de réparations si celles-ci sont permises.

Nous avons passé en revue les éléments de coloration de graphes qui seront à l'œuvre dans les Chapitres 3 à 5, mais tel que mentionné plus haut il s'agit d'un domaine de recherche à part entière très fécond comprenant de nombreux autres aspects qui ne sont pas utilisés ici et donc pas décrits dans cette revue de littérature. Multicoloration, coloration orientée, coloration décentralisée, coloration des arêtes sont des exemples de tels thèmes que peuvent approfondir les lecteurs particulièrement curieux.

CHAPITRE 3 RÉSOLUTION EXACTE PAR PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS

Cette section est dédiée à la résolution exacte du problème, qui à partir de maintenant est noté le (k, W, ω, θ)-Plus Grand Sous-graphe Induit (PGSI) colorable. Trois formulations en nombres entiers sont proposées pour calculer le paramètre $\alpha_k^\theta(G, W, \omega)$, dont une basée sur la décomposition de Dantzig-Wolfe. Celles-ci seront analysées et améliorées au moyen d'inégalités valides et de coupes s'appuyant sur les problèmes classiques de sac-à-dos binaire et de pavage d'ensemble. Une borne supérieure sur l'objectif développée à partir de la fonction généralisée de Lovász est également proposée. Nous verrons que celle-ci s'avère plus serrée que la relaxation linéaire des deux premières formulations. Des simulations numériques mettront en évidence dans quelle mesure ces éléments permettent d'accélérer les temps de résolution, et quelles stratégies sont les plus efficaces. Dans les meilleurs cas, plusieurs milliers de nœuds de branchement peuvent être éliminés. L'objectif de cette section est double : explorer la nature combinatoire du problème, et déterminer une manière efficace de résoudre le problème de façon exacte, de façon à avoir une référence pour les heuristiques développées dans le chapitre suivant.

3.1 Modèles en nombres entiers

Formulation I

La première formulation est inspirée de celle de Araujo et al. [46]. Soit x_{vi} une variable binaire égale à 1 si et seulement si le sommet $v \in V$ prend la couleur $i \in \{1, \cdots, k\}$. On peut alors obtenir $\alpha_k^\theta(G, W, \omega)$ à l'aide du modèle suivant :

$$\max_x \sum_{v \in V} \sum_{i=1}^{k} x_{vi} \tag{3.1}$$

$$\text{sujet à} \quad \sum_{i=1}^{k} x_{vi} \leq 1 \qquad \forall v \in V \tag{3.2}$$

$$\sum_{u \neq v} \omega(u, v) \cdot x_{ui} \leq \theta \cdot W(v) + M_v \cdot (1 - x_{vi}) \qquad \forall v \in V, \forall i \in \{1, \cdots, k\} \tag{3.3}$$

$$x_{vi} \in \{0, 1\} \qquad \forall v \in V, \forall i \in \{1, \cdots, k\} \tag{3.4}$$

L'objectif (3.1) vise à maximiser le nombre de sommets colorés. Les contraintes (3.2) assurent qu'au plus une couleur soit affectée à chaque sommet. Enfin, l'équation (3.3) est une

contrainte disjonctive qui impose que la coloration obtenue soit bien θ-impropre ; M_v est une constante suffisamment grande, par exemple $M_v = \sum_{u \neq v} \omega(u,v) - \theta \cdot W(v)$ suffit.

Formulation II

Le modèle précédent peut être légèrement amélioré de la façon suivante. Soit y^i_{uv} un autre ensemble de variables binaires prenant la valeur 1 si et seulement si les sommets u et v sont simultanément affectés de la couleur $i \in \{1, ..., k\}$. Le modèle s'écrit alors comme suit :

$$\max_{x,y} \sum_{v \in V} \sum_{i=1}^{k} x_{vi} \tag{3.5}$$

$$\text{sujet à} \quad \sum_{i=1}^{k} x_{vi} \leq 1 \qquad \forall v \in V \tag{3.6}$$

$$\sum_{u \neq v} \omega(u,v) \cdot y^i_{uv} \leq \theta \cdot W(v) \cdot x_{vi} \qquad \forall v \in V, \forall i \in \{1, \cdots, k\} \tag{3.7}$$

$$x_{ui} + x_{vi} \leq y^i_{uv} + 1 \qquad \forall u \neq v, \forall i \in \{1, \cdots, k\} \tag{3.8}$$

$$x_{vi},\ y^i_{uv} \in \{0,1\} \qquad \forall u \neq v, \forall i \in \{1, \cdots, k\} \tag{3.9}$$

Les contraintes (3.8) lient les variables y^i_{uv} et x^i_v entre elles, imposant à y^i_{uv} de prendre la valeur 1 si u et v ont la même couleur i. Bien sûr, $y^i_{uv} = y^i_{vu}$ pour toute paire de sommets $\{u, v\}$. Les autres contraintes sont identiques que précédemment.

Formulation III

La troisième formulation est une décomposition de Dantzig-Wolfe de la formulation II, donnant lieu à un problème dit de *pavage* d'ensemble. Cette formulation est basée sur le changement de variables suivant. Soit Ω_i l'ensemble des points extrêmes de l'enveloppe convexe définie par les équations (3.7) - (3.9) pour la couleur i ; autrement dit, Ω_i est l'ensemble de tous les sous-ensembles θ-indépendants de G pouvant prendre la couleur i. Soit λ_{ip} une variable binaire égale à 1 si et seulement si le sous-ensemble $p \in \Omega$ est selectionné, et prend la couleur i. Il est alors possible d'exprimer chaque variable x_{vi} comme combinaison convexe

de ces points extrêmes [55] :

$$x_{vi} = \sum_{p \in \Omega_i | v \in p} \lambda_{ip} \qquad \forall v \in V, \forall i \in \{1, \cdots, k\}$$

$$\sum_{p \in \Omega_i} \lambda_{ip} = 1 \qquad \forall i \in \{1, \cdots, k\}$$

$$\lambda_{ip} \geq 0 \qquad \forall p \in \Omega_i, \forall i \in \{1, \cdots, k\}$$

En substituant x_{vi} dans la fonction objectif (3.5) et dans les contraintes (3.6), on obtient la formulation suivante :

$$\max_{\lambda} \sum_{v \in V} \sum_{i=1}^{k} \sum_{p \in \Omega_i | v \in p} \lambda_{ip} \tag{3.10}$$

sujet à :

$$\sum_{i=1}^{k} \sum_{p \in \Omega_i | v \in p} \lambda_{ip} \leq 1 \qquad \forall v \in V \tag{3.11}$$

$$\sum_{p \in \Omega_i} \lambda_{ip} = 1 \qquad \forall i \in \{1, \cdots, k\} \tag{3.12}$$

$$\lambda_{ip} \geq 0 \qquad \forall p \in \Omega_i, \forall i \in \{1, \cdots, k\} \tag{3.13}$$

$$\sum_{p \in \Omega_i | v \in p} \lambda_{ip} \in \{0, 1\} \qquad \forall v \in V, \forall i \in \{1, \cdots, k\} \tag{3.14}$$

Tel qu'expliqué dans [96], il est possible de simplifier ce modèle. Premièrement, les variables x_{vi} étant binaires, tous les points entiers de l'enveloppe convexe définie par les équations (3.7) - (3.9) sont des points extrêmes, et l'on peut donc remplacer les contraintes (3.14) par

$$\lambda_{ip} \in \{0, 1\} \ \forall p \in \Omega_i, \ \forall i \in \{1, \cdots, k\}.$$

De plus, puisque l'ensemble vide est un point extrême de coût nul, il peut être omis de Ω_i et la contrainte de convexité (3.12) est remplacée par l'équation

$$\sum_{p \in \Omega_i} \lambda_{ip} \leq 1 \ \forall i \in \{1, \cdots, k\}.$$

Toutes les couleurs étant équivalentes, il est judicieux d'agréger les variables de façon à

éliminer des solutions symétriques en posant

$$\lambda_p = \sum_{i=1}^{k} \lambda_{ip} \ \in \{0,1\} \quad \forall p \in \Omega; \tag{3.15}$$

Ω représente alors l'ensemble de tous les sous-ensembles θ-independants de G, et λ_p est une variable binaire égale à 1 si et seulement si le sous-ensemble $p \in \Omega$ est selectionné. La contrainte de convexité (3.12) s'écrit alors

$$\sum_{p \in \Omega} \lambda_p \leq k, \tag{3.16}$$

et le modèle final s'écrit :

$$\max_{\lambda} \sum_{v \in V} \sum_{p \in \Omega | v \in p} \lambda_p \tag{3.17}$$

$$\text{sujet à} \quad \sum_{p \in \Omega | v \in p} \lambda_p \leq 1 \qquad \forall v \in V \tag{3.18}$$

$$\sum_{p \in \Omega} \lambda_p \leq k \tag{3.19}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in \Omega \tag{3.20}$$

La fonction objectif (3.17) maximise le nombre de sommets sélectionnés parmi les sous-ensembles θ-indépendants. Les contraintes (3.18) imposent qu'un sommet soit coloré au plus une fois, et les contraintes (3.19) limitent le nombre de sous-ensembles θ-indépendants (et donc le nombre de couleurs) à k.

Comparaison des relaxations continues

Proposition 6. *Soient X^I_{RL}, X^{II}_{RL} et X^{III}_{RL} les polytopes définis par les relaxations continues des formulations I, II and III respectivement (i.e., les contraintes d'intégralité (3.4), (3.9) et (3.20) sont omises). Alors, il existe $\phi_1 : X^{III}_{RL} \to X^{II}_{RL}$ et $\phi_2 : X^{II}_{RL} \to X^{I}_{RL}$ telles que*

i. $\mathrm{Im}_{\phi_1}(X^{III}_{RL}) \subseteq X^{II}_{RL}$,

ii. $X^{II}_{RL} \neq \mathrm{Im}_{\phi_1}(X^{III}_{RL})$,

iii. $\mathrm{Im}_{\phi_2}(X^{II}_{RL}) \subseteq X^{I}_{RL}$,

iv. $X^{I}_{RL} \neq \mathrm{Im}_{\phi_2}(X^{II}_{RL})$.

Démonstration. i. Soit $(\hat{x}_{uv}, \hat{y}^i_{uv}) \in X^{II}_{RL}$. En combinant les équations (3.7) et (3.8), on

obtient :

$$\sum_{u \neq v} \omega(u,v) \cdot \hat{x}_{ui} \leq \sum_{u \neq v} \omega(u,v) + \hat{x}_{vi} \left(\theta \cdot W(v) - \sum_{u \neq v} \omega(u,v) \right).$$

En posant $M_v = \sum_{u \neq v} \omega(u,v) - \theta \cdot W(v)$, en additionant et soustrayant $\theta \cdot W(v)$ sur le membre de droite, on a alors :

$$\sum_{u \neq v} \omega(u,v) \cdot \hat{x}_{ui} \leq \theta \cdot W(v) + M_v \cdot (1 - \hat{x}_{vi}),$$

ce qui correspond exactement à l'équation (3.3). En notant $\phi_2 : X^{II}_{RL} \rightarrow X^{I}_{RL}$ la transformation requise pour aboutir à cette inégalité, on a bien $\phi_2(\hat{x}_{uv}, \hat{y}^i_{uv}) \in X^{I}_{RL}$.
Soit $\hat{\lambda}_p \in X^{III}_{RL}$. En posant $x_{vi} = \hat{\lambda}_p$ pour tout sommet v de l'ensemble $p \in \Omega$, les equations (3.6)-(3.8) sont satisfaites. En effet, les contraintes (3.6) et (3.18) sont alors équivalentes, et les contraintes (3.6)-(3.7) sont satisfaites par définition des variables λ_p, ce qui prouve qu'il existe $\phi_1 : X^{III}_{RL} \rightarrow X^{II}_{RL}$ telle que $\phi_1(\hat{\lambda}_p) \in X^{II}_{RL}$.

ii. Pour montrer que $X^{I}_{RL} \neq \mathrm{Im}_{\phi_2}(X^{II}_{RL})$, considérons le contre-exemple illustré à la Figure 3.1 (à gauche), où $k = \theta = 1$. La solution suivante est réalisable pour la formulation I (sans les contraintes d'intégralité) : $x_{a1} = 1$ et $x_{b1} = x_{c1} = 0.4$, mais non réalisable pour la formulation II : les contraintes (3.7) sont violées en b (et c). En effet, de telles valeurs imposent $y^1_{ab} \geq 0.4$, $y^1_{ac} \geq 0.4$ et $y^1_{bc} \geq 0$, et donc $\omega(a,b) \cdot y^1_{ab} + \omega(c,b) \cdot y^1_{bc} \geq 2 \cdot 0.4 + 2 \cdot 0 = 0.8 > 0.4 = \theta \cdot W(b) \cdot x_{b1}$.

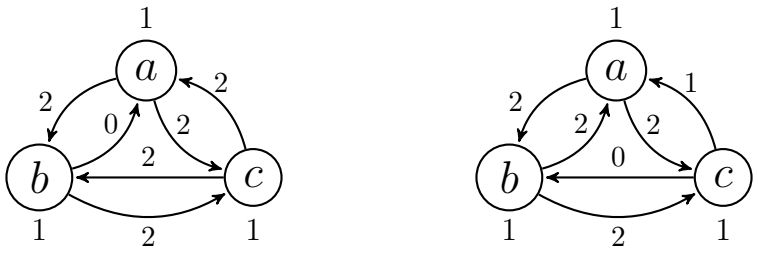


Figure 3.1 Exemples où $X^{I}_{RL} \neq \mathrm{Im}_{\phi_2}(X^{II}_{RL})$ et $X^{II}_{RL} \neq \mathrm{Im}_{\phi_1}(X^{III}_{RL})$

Pour montrer que $X^{II}_{RL} \neq \mathrm{Im}_{\phi_1}(X^{III}_{RL})$, considérons le contre-exemple illustré à la Figure 3.1 (à droite) avec $k = \theta = 1$. La solution suivante est réalisable pour la formulation II (sans les contraintes d'intégralité) : $x_{a1} = x_{c1} = 0.5$, $x_{b1} = 0.75$, $y^1_{ab} = y^1_{bc} = 0.25$ et $y^1_{ac} = 0$. Autrement dit l'objectif $x_{a1} + x_{b1} + x_{c1}$ est strictement plus grand que 1, mais ceci est impossible avec la formulation III, puisque les seuls ensembles θ-indépendants non vides sont les singletons $\{a\}$, $\{b\}$, $\{c\}$.

□

On peut s'attendre avec ces inclusions à ce que la formulation I donne les pires sauts d'intégralité, et la formulation III les meilleurs. Les tests numériques ci-après confirmeront cela. Cependant, puisque le nombre de sous ensembles θ-indépendants peut s'élever à $\mathcal{O}(2^n)$, la formulation III utilise un nombre exponentiel de variables et peut donc difficilement être utilisée telle quelle. Nous verrons comment générer uniquement les variables prometteuses pour ce modèle.

3.2 Bornes sur l'objectif

Bien souvent, l'ajout de bornes sur l'objectif permet d'accélérer la résolution des problèmes en nombres entiers. Tel que décrit à la Section 2.5.2, les travaux de Narasimhan [53] sont consacrés à l'étude de $\alpha_k(G)$, on y trouve justement une borne supérieure sur ce paramètre basée sur le fonction généralisée de Lovász :

$$\alpha_k(G) \leq \vartheta_k(\bar{G}).$$

Voyons comment l'adapter au paramètre $\alpha_k^\theta(G, W, \omega)$ sur des graphes complets, orientés, pondérés. Pour se replacer dans une configuration non orientée, introduisons la définition suivante.

Définition 23. *Pour tout graphe $G(V, W, \omega)$ orienté, complet, pondéré, on considère le graphe auxiliaire $H(V, E)$, où :*

i. L'ensemble des sommets de H est une copie de celui de G,

ii. Une arête $\{u, v\}$ est créée entre u et v si et seulement si $\omega(u, v) > \theta \cdot W(v)$ ou $\omega(v, u) > \theta \cdot W(u)$.

Autrement dit, il existe une arête entre deux sommets u et v si aucune k-coloration θ-impropre dans laquelle u et v ont la même couleur peut être obtenue, et le graphe auxiliaire H est défini de sorte que toute k-coloration θ-impropre de G est une k-coloration de H. La borne supérieure sur l'objectif prend alors la forme suivante.

Proposition 7. *Considérons le graphe auxiliaire H de la Définition 23. Alors :*

$$\alpha_k^\theta(G, W, \omega) \leq \left\lfloor \vartheta_k(\bar{H}) \right\rfloor. \tag{3.21}$$

Démonstration. La preuve découle du fait que toute k-coloration θ-impropre de G est une k-coloration de H, par construction. Par conséquent, d'une part on a $\alpha_k^\theta(G, W, \omega) \leq \alpha_k(H)$, d'autre part $\alpha_k(H) \leq \vartheta_k(\bar{H})$. □

Considérons l'exemple illustré à la Figure 3.2, avec $k = \theta = 2$. Le graphe initial est à gauche, le graphe auxiliaire au milieu, et son complémentaire à droite. On a vu dans l'exemple 2.15 que $\vartheta_2(C_5) = 2\sqrt{5}$. Par conséquent $\alpha_2^2(G, W, \omega) \leq \left\lfloor 2\sqrt{5} \right\rfloor = 4$; cette borne est atteinte en colorant les sommets a et b d'une couleur, et les sommets c et d de l'autre.

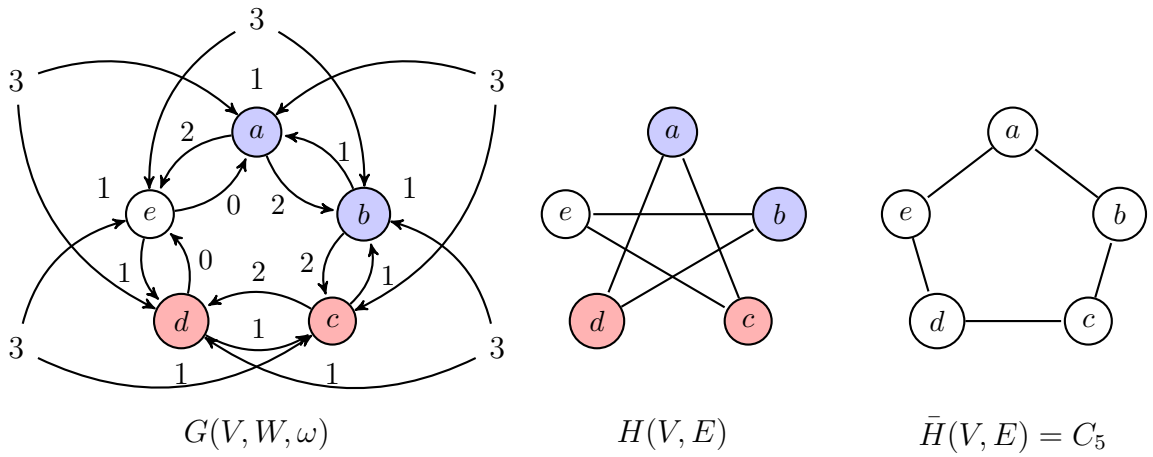


Figure 3.2 Un exemple : $\alpha_2^2(G, W, \omega) \leq \left\lfloor \vartheta_2(\bar{H}) \right\rfloor = \left\lfloor 2\sqrt{5} \right\rfloor = 4$

Dans cet exemple, la borne est atteinte et la 2-coloration de H est une 2-coloration 2-impropre de G. Mais de manière générale, une k-coloration de H ne correspond pas nécessairement à une k-coloration θ-impropre de G, et l'écart entre $\vartheta_k(\bar{H})$ et $\alpha_k^\theta(G, W, \omega)$ peut être arbitrairement grand. Si par exemple $\theta = W(v) = 1$ pour tout sommet v, et $\omega(u, v) > \frac{1}{2}$ pour tout arc (u, v), alors il est clair que $\alpha_k^1(G, W, \omega) = 2k$ (on ne peut colorer que des paires de sommets de la même couleur). Mais $\bar{H}$ est alors un graphe complet d'ordre n, donc $\vartheta_k(\bar{H}) = n$ (l'ensemble $\mathcal{A}$ est constitué d'une seule matrice carré d'ordre n, remplie de 1), et il s'ensuit que le ratio $\frac{\vartheta_k(\bar{H})}{\alpha_k^\theta(G,W,\omega)} = \frac{n}{2k}$ peut être arbitrairement grand.

3.3 Résolution par branch-and-cut

Nous avons vu que les algorithmes de plans coupants (et donc de branch-and-cut) consistent à redéfinir itérativement le polyèdre, jusqu'à ce qu'un point extrême corresponde à une solution entière. L'objectif est donc d'ajouter des inégalités de façon à approcher l'enveloppe convexe définie par les contraintes du problème. Ces inégalités peuvent être issues de considérations logiques et sont ajoutées préalablement à la résolution, ou elles peuvent être construites à partir d'une solution fractionnaire, en cours du processus de résolution. Voyons plus en détail quels types d'inégalités peuvent s'appliquer à notre problème.

3.3.1 Inégalités valides

Contraintes d'asymétrie

Une des difficultés des problèmes de coloration est le fait que l'espace de recherche contient de nombreuses solutions (fractionnaires) équivalentes où les couleurs sont simplement permutées. Pour briser cette symétrie, on peut ajouter au modèle (formulations I ou II) les contraintes suivantes :

$$\sum_{u \in V} x_{ui} \geq x_{v(i+1)} \qquad \forall v \in V,\ \forall i \in \{1, \cdots, k-1\} \tag{3.22}$$

$$\sum_{u \in V} x_{ui} \geq \sum_{u \in V} x_{u(i+1)} \qquad \forall i \in \{1, \cdots, k-1\} \tag{3.23}$$

L'équation (3.22) empêche que la couleur $i+1$ ne soit utilisée tant que la couleur i ne l'est pas, et l'équation (3.23) trie les classes de couleur par cardinalité non croissante. Ces contraintes d'asymétrie ordonnent les solutions réalisables dans l'espace de recherche et permettent de supprimer des régions inutiles du polyèdre.

Contraintes d'incompatibilité

Reconsidérons le graphe auxiliaire $H = (V, E)$ défini plus haut. Si $\{u, v\} \in E$, alors u et v ne peuvent pas prendre simultanément la même couleur, ce qui peut être imposé comme suit (pour la formulation II) :

$$y_{uv}^{i} = 0 \quad \forall \{u, v\} \in E,\ \forall i \in \{1, \cdots, k\}. \tag{3.24}$$

Ces contraintes peuvent être vues comme un prétraitement du problème où sont éliminées des solutions impossibles évidentes ; il suffit de parcourir l'ensemble des arcs en un temps $\mathcal{O}(n^2)$ pour les identifier.

Les inégalités valides qui suivent sont décrites pour la formulation II, mais peuvent être aisément adaptées à la formulation I. Il s'agit de contraintes supplémentaires que l'on peut ajouter au problème au cours du processus de résolution, afin d'éliminer une solution fractionnaire du polyèdre.

3.3.2 Coupes de sac-à-dos binaire

Dans la formulation II, la coloration θ-impropre est assurée grâce aux contraintes

$$\sum_{u \neq v} \omega(u,v) \cdot y_{uv}^{i} \leq \theta \cdot W(v) \cdot x_{vi} \quad \forall v \in V,\ \forall i \in \{1, \cdots, k\}.$$

Ainsi, si x_{vi} prend la valeur 0, alors y_{uv}^{i} est également nulle. Mais si x_{vi} prend la valeur 1 (si la couleur i est attribuée au sommet v), cette inéquation devient une contrainte d'un problème de sac-à-dos binaire associé au couple (v, i), où les sommets sont des objets de coût unitaire et de poids $\omega(u, v)$ que l'on veut faire rentrer dans un sac de capacité $\theta \cdot W(v)$. On peut alors exploiter les coupes déjà existantes pour ce problème. On rappelle les résultats suivants.

Proposition 8. *Considérons le problème de sac-à-dos binaire suivant. On dispose de n objets de poids v_i, et d'un sac de capacité donnée. Soit x_i une variable binaire égale à 1 si l'on sélectionne l'objet i, soit C un sous-ensemble d'objets qui excèdent la capacité du sac, et soit P un sous-ensemble d'objets qui y rentrent en laissant une capacité résiduelle égale à μ. Alors, les inégalités suivantes sont valides :*

i. $\sum_{i \in C} x_i \leq |C| - 1,$

ii. $\sum_{i \notin P} \max\{0, v_i - \mu\} \cdot x_i \leq \sum_{i \in P} v_i \cdot (1 - x_i).$

Ces inégalités sont appelées coupes de recouvrement (i.) et de poids (ii.). Le lecteur peut se référer à [97] pour plus de détails.

Pour adapter ces inégalités à notre problème, on introduit les définitions suivantes.

Définition 24. *Soit un graphe orienté, complet, pondéré, $G = (V, W, \omega)$. Une couverture d'un sommet v, notée $C(v)$, est un sous-ensemble de V tel que*

$$\sum_{u \in C(v)} \omega(u,v) > \theta \cdot W(v),$$

et un paquet d'un sommet v, noté $P(v)$, est un sous-ensemble de sommets de V tel que

$$\sum_{u \in P(v)} \omega(u,v) \leq \theta \cdot W(v).$$

La capacité résiduelle ($\theta \cdot W(v) - \sum_{u \in P(v)} \omega(u,v)$) est notée μ_v.

On en déduit les coupes suivantes pour la formulation II.

Corollaire 1 (Coupes de recouvrement)**.** *Toute variable binaire y^i_{uv} doit vérifier l'inégalité suivante :*

$$\sum_{u \in C(v)} y^i_{uv} \leq |C(v)| - 1 \quad \forall C(v),\ \forall i \in \{1, \cdots, k\}. \tag{3.25}$$

Essentiellement, ces contraintes imposent que les sommets de $C(v)$ ne puissent pas prendre simultanément la même couleur que v. Elles sont violées par une solution fractionnaire $\hat{y}^i_{uv}$ si

$$\sum_{u \in C(v)} (1 - \hat{y}^i_{uv}) < 1.$$

Tel que décrit dans [97], on peut générer de telles coupes de façon heuristique à partir d'une solution fractionnaire $\hat{y}^i_{uv}$, en insérant les sommets dans l'ensemble $C(v)$ par ordre de $\frac{1-\hat{y}^i_{uv}}{\omega(u,v)}$ non décroissant. L'idée est de donner la priorité aux sommets qui ont un coefficient $1 - \hat{y}^i_{uv}$ le plus petit possible, de façon à ce que l'inégalité soit bel et bien violée, tout en favorisant l'insertion de sommets susceptibles de former une couverture, c'est-à-dire dont le poids $\omega(u, v)$ est aussi suffisamment grand.

Corollaire 2 (Coupes de poids)**.** *Toute variable binaire y^i_{uv} doit vérifier l'inégalité suivante :*

$$\sum_{u \notin P(v)} \max\{0, \omega(u,v) - \mu_v\} \cdot y^i_{uv} \leq \sum_{u \in P(v)} \omega(u,v) \cdot \left(1 - y^i_{uv}\right) \quad \forall P(v),\ \forall i \in \{1, \cdots, k\}. \tag{3.26}$$

Tel que décrit dans [98], on peut générer de telles coupes de façon heuristique à partir d'une solution fractionnaire $\hat{y}^i_{uv}$, en insérant dans $P(v)$ de façon gloutonne les sommets dont le coefficient $\hat{y}^i_{uv}$ est le plus proche de 1, et dont le poids $\omega(u, v)$ est le plus faible possible. L'idée est de sélectionner un ensemble de sommets susceptibles de violer l'inégalité (c'est-à-dire tel que le membre de droite soit le plus petit possible) tout en formant un paquet.

3.3.3 Coupes de recouvrement

Une autre stratégie possible consiste à s'inspirer des coupes existantes pour le problème du stable maximum, qui est un cas particulier de notre problème : il suffit de poser $k = 1$, $\theta = 0$, $W(v) = 1$ pour tout sommet v, et $\omega(u, v) = 1$ pour tout arc (u, v). Il s'agit de trouver le plus grand sous graphe induit, de sorte que deux sommets ne soient pas directement voisins, autrement dit c'est un problème de pavage d'ensemble pour lequel des coupes génériques ont été largement étudiées [99, 100, 101]. Commençons par rappeler certains de ces résultats, qui

serviront de base pour de nouvelles inégalités valides ici.

Proposition 9. *Soit $G = (V, E)$ un graphe. Considérons les variables binaires x_u égales à 1 si et seulement si le sommet u est sélectionné. Alors, les inégalités suivantes sont valides pour le problème du stable maximum de G :*

i. $\sum_{u \in K} x_u \leq 1$ *pour toute clique K de G,*

ii. $\sum_{u \in C} x_u \leq \frac{|C|-1}{2}$ *pour tout cycle impair C qui est un sous-graphe partiel de G.*

Ces inégalités sont appelées inégalités de clique (i.) et de cycle impair (ii.). Le lecteur peut se référer à [100] pour plus de détails.

En reprenant le graphe auxiliaire de la Définition 23, les corollaires suivants sont triviaux.

Corollaire 3 (Inégalités de clique)**.** *Soit K une clique du graphe auxiliaire H. Alors, les inégalités suivantes sont valides pour la formulation II :*

$$\sum_{v \in K} x_{vi} \leq 1 \quad \forall i \in \{1, \cdots, k\}. \tag{3.27}$$

Ceci garantit que les sommets ne pouvant être simultanément affectés de la même couleur soient bien isolés. C'est en quelque sorte une extension des inégalités d'incompatibilité (3.24) décrites plus haut. Cette inégalité est violée par une solution fractionnaire $\hat{x}_{vi}$ si

$$\sum_{v \in K} \hat{x}_{vi} > 1,$$

on peut donc espérer l'identifier en sélectionnant de façon gloutonne les variables $\hat{x}_{vi}$ qui prennent la plus grande valeur, et dont le sommet v est adjacent aux sommets précédemment séléctionnés, jusqu'à ce qu'éventuellement $\sum_{v \in K} \hat{x}_{vi} > 1$.

Corollaire 4 (Inégalités de cycle impair)**.** *Soit C un cycle impair qui est un sous-graphe partiel dans le graphe auxiliaire H. Alors, les inégalités suivantes sont valides pour la formulation II :*

$$\sum_{v \in C} x_{vi} \leq \frac{|C|-1}{2} \quad \forall i \in \{1, \cdots, k\}. \tag{3.28}$$

Cette inégalité est violée par une solution fractionnaire $\hat{x}_{vi}$ si

$$\sum_{v \in C} (1 - 2\hat{x}_{vi}) < 1,$$

et peut être identifiée de la façon suivante [101] : pour chaque couleur i, considérons le graphe biparti $B = (V_1 \cup V_2, E)$, où $V_1 = V_2 = V$, et E est l'ensemble des arêtes de B tel que $u \in V_1$ et $v \in V_2$ sont reliés par une arête de poids $1 - \hat{x}_{ui} - \hat{x}_{vi}$ si les sommets correspondants dans le graphe H sont adjacents. Chaque sommet $v_j \in V_1$ a donc une copie $u_j \in V_2$. À partir de cette construction, on cherche un plus court chemin dans B entre un sommet $u_0 \in V_1$ et sa copie $v_0 \in V_2$. L'idée est de trouver un chemin de la forme $(u_0, v_1, u_2, \cdots, u_{2j}, v_0)$, qui correspond au cycle impair $(u_0, u_1, \cdots, u_{2j}, u_0)$ dans H. Si le plus court chemin trouvé est de longueur inférieure à 1, l'inégalité est violée par les variables fractionnaires des sommets du chemin. Remarquons qu'avec les contraintes d'incompatibilité (3.24), les poids sont nécessairement positifs, et l'on peut déterminer ce plus court chemin avec un algorithme de Dijkstra [102] entre un sommet $u_0 \in V_1$ et sa copie $v_0 \in V_2$.

3.4 Résolution par branch-and-price

Dans cette section, nous décrivons comment résoudre le modèle de la formulation III (équations (3.17)-(3.20)) à l'aide d'une procédure dite de génération de colonnes, intégrée dans un algorithme de branch-and-bound. Tel que mentionné plus haut, cette stratégie a été utilisée dans [3] et [58] pour calculer $\chi_\theta(G, W, \omega)$ et $\alpha_0^1(G, 1, 1)$, respectivement.

Nous avons vu que la formulation III est attrayante dans la mesure où le polyèdre des contraintes est plus serré que celui des formulations I et II, et donc la probabilité d'obtenir une solution entière en relâchant les contraintes d'intégralité est plus élevée. En revanche, le nombre de variables est exponentiel, il convient donc de ne pas toutes les considérer mais de générer dynamiquement uniquement celles qui sont prometteuses. Tel est le but de la génération de colonnes.

3.4.1 Problème maître et sous-problème

On appelle *problème maître restreint* (PMR) la relaxation continue de la formulation III :

$$\max_{\lambda} \sum_{v \in V} \sum_{p \in \Omega | v \in p} \lambda_p \tag{3.29}$$

$$\text{sujet à} \quad \sum_{p \in \Omega | v \in p} \lambda_p \leq 1 \qquad \forall v \in V \tag{3.30}$$

$$\sum_{p \in \Omega} \lambda_p \leq k \tag{3.31}$$

$$\lambda_p \geq 0 \qquad \forall p \in \Omega \tag{3.32}$$

Et l'on appelle *sous-problème* le problème qui consiste à déterminer les variables λ_p susceptibles de donner lieu à de bonnes solutions du PMR. « Bonnes solutions » est ici synonyme de solutions en mesure d'améliorer la fonction objectif, c'est-à-dire dont le coût réduit est strictement positif. En notant π_v et γ les variables duales associées aux contraintes (3.30) et (3.31) du PMR, le coût réduit d'une variable λ_p a pour expression $\sum_{v \in V | v \in p}(1 - \pi_v) - \gamma$. En introduisant une variable binaire x_v égale à 1 si et seulement si le sommet v est sélectionné pour faire partie d'un ensemble θ-indépendant $p \in \Omega$, le sous-problème peut s'écrire de la façon suivante :

$$\max_x \sum_{v \in V} (1 - \pi_v) x_v - \gamma \tag{3.33}$$

$$\text{sujet à} \quad \sum_{u \neq v} \omega(u, v)\, y_{uv} \leq \theta \cdot W(v) \cdot x_v \qquad \forall v \in V \tag{3.34}$$

$$x_u + x_v \leq y_{uv} + 1 \qquad \forall u \neq v \tag{3.35}$$

$$x_v \in \{0, 1\} \qquad \forall v \in V \tag{3.36}$$

On reconnaît dans la fonction objectif le coût réduit d'une variable λ_p, et dans les contraintes des équations analogues à celles de la formulation II, qui garantissent que les sommets sélectionnés forment bel et bien un ensemble θ-indépendant.

3.4.2 Génération de colonnes

Le principe de la génération de colonnes est le suivant : le problème maître restreint et le sous-problème sont résolus itérativement jusqu'à ce qu'il n'y ait plus de colonne - ou variable λ_p - de coût réduit positif pouvant améliorer la qualité de la solution optimale du problème maître restreint. Ce processus est résumé dans les Tableaux 3.1 -3.3 ci-après. On obtient en bout de ligne une borne supérieure sur le paramètre $\alpha_k^\theta(G, W, \omega)$, à partir de laquelle on peut exécuter une itération de branch-and-bound.

Tableau 3.1 Résolution du PMR

Algorithme 3: MasterSolve

ENTRÉE :
Un ensemble Ω de variables λ_p représentant des ensembles θ-indépendants ;
RÉSULTAT :
- La solution $\bar{z}$ du PMR (3.29) - (3.31);
- Un ensemble de variables duales (π_v, γ) associées aux contraintes (3.30) et (3.31).

Tableau 3.2 Résolution du sous-problème

Algorithme 4: SubSolve

ENTRÉE :
Un ensemble de variables duales associées aux contraintes (3.30) - (3.31) :(π_v, γ) ;
RÉSULTAT :
$\begin{cases} \text{Une colonne } \lambda_p \text{ de coût réduit positif représentant un ensemble } \theta\text{-indépendant } S_\theta; \\ \text{OU} \quad \varnothing. \end{cases}$

Tableau 3.3 Procédure de génération de colonnes

Algorithme 5: Génération de colonnes

$\Omega = \varnothing$;
tant que $SubSolve(\pi_v, \gamma) \neq \varnothing$ **faire**
    $(\bar{z}, \pi_v, \gamma) = MasterSolve(\Omega)$
    $S_\theta = SubSolve(\pi_v, \gamma)$
    $\Omega \leftarrow \Omega \cup \{S_\theta\}$
retourner $\bar{z}$

En pratique, avant que l'ensemble Ω ne soit suffisamment riche, $\bar{z}$ est une borne inférieure sur $\alpha_k^\theta(G, W, \omega)$. C'est une des raisons pour lesquelles certains préconisent de résoudre le dual du problème maître, de façon à avoir à chaque itération une borne supérieure. Ce dual n'est rien d'autre que la relaxation Lagrangienne du problème (formulation II). Voyons cela plus en détail.

3.4.3 Lien avec la relaxation Lagrangienne

La relaxation Lagrangienne de la formulation II par rapport aux contraintes (3.6) s'écrit de la façon suivante :

$$L(\pi_v) = \max_x \sum_{v \in V} \sum_{i=1}^{k} x_{vi} - \sum_{v \in V} \pi_v \left(\sum_{i=1}^{k} x_{vi} - 1 \right) \tag{3.37}$$

$$\text{sujet à} \quad \sum_{u \neq v} \omega(u,v) \cdot x_{ui} \leq \theta \cdot W(v) \cdot x_{vi} \qquad \forall v \in V, \forall i \in \{1, ..., k\} \tag{3.38}$$

$$x_{ui} + x_{vi} \leq y_{uv}^i + 1 \qquad \forall u \neq v, \forall i \in \{1, \cdots, k\} \tag{3.39}$$

$$x_{vi},\ y_{uv}^i \in \{0, 1\} \qquad \forall u \neq v, \forall i \in \{1, \cdots, k\} \tag{3.40}$$

où $\pi_v \in \mathbb{R}^+$ est la pénalité encourue si la contrainte (3.6) est violée pour le sommet v. Pour des valeurs fixées de π_v, la solution à ce problème donne une borne supérieure sur $\alpha_k^\theta(G, W, \omega)$. En faisant apparaître la constante γ et en réordonnant les termes, l'objectif peut se réécrire comme suit :

$$L(\pi_v) = \sum_{v \in V} \pi_v + \gamma + \max_x \sum_{v \in V} \sum_{i=1}^{k} (1 - \pi_v) x_{vi} - \gamma$$

Par le théorème de dualité forte on peut remplacer la somme $\sum_{v \in V} \pi_v + \gamma$ par la valeur de l'objectif du PMR, notée $\underline{z}$. Et puisque le problème est séparable par couleur, l'objectif se réécrit finalement

$$L(\pi_v) = \underline{z} + k \left(\max_x \sum_{v \in V} (1 - \pi_v) x_v - \gamma \right).$$

On reconnaît alors l'objectif du sous-problème, aux coefficients $\underline{z}$ et k près. Moralité, à toute solution du sous-problème correspond une borne supérieure sur le problème, appelée borne Lagrangienne.

Proposition 10 (Borne Lagrangienne)**.** *Soit $\hat{c}^* = \sum_{v \in V} (1 - \pi_v) x_v^* - \gamma$ la solution optimale du sous-problème à une itération donnée, soit $\underline{z}$ la solution courante du PMR, et soit $\bar{z}$ la borne supérieure obtenue par relaxation Lagrangienne des contraintes (3.6). À toute itération du processus de génération de colonnes, on a l'égalité*

$$\bar{z} = \underline{z} + k\hat{c}^*.$$

On peut interpréter cette égalité par le fait qu'à chaque itération, la solution courante du PMR $\underline{z}$ peut être améliorée par au plus k fois la solution donnée par le sous-problème. Une fois que le sous-problème ne trouve plus de colonne, on a $\hat{c}^* = 0$, et par conséquent $\bar{z} = \underline{z}$. Relaxation Lagrangienne et génération de colonnes sont donc deux méthodes permettant d'obtenir la même borne supérieure sur le paramètre $\alpha_k^\theta(G, W, \omega)$. Les deux méthodes résolvent à chaque itération le même sous-problème, l'une résout le primal du problème maître, et l'autre son dual.

3.4.4 Stratégies d'accélération

Si la décomposition de Dantzig-Wolfe est très attrayante sur le plan théorique, sa mise œuvre est bien souvent difficile, notamment parce qu'à chaque itération, il faut résoudre de nombreux problèmes en nombres entiers. Voici une série de stratégies permettant de gagner en vitesse d'exécution.

Agrégation des variables

Nous avons vu que les couleurs jouant un rôle équivalent, il est possible d'agréger les variables $\sum_{i=1}^{k} \lambda_{ip}$ (voir l'équation 3.15). Il n'est donc pas nécessaire de résoudre un sous-problème par couleur : une seule résolution suffit.

Heuristique pour le sous-problème

La résolution optimale du sous-problème à chaque itération peut s'avérer coûteuse et inutile. On peut préférer le résoudre à l'aide d'une heuristique, jusqu'à ce que celle-ci n'y parvienne plus. Le sous-problème peut être vu comme la recherche de l'ensemble θ-indépendant de poids maximum, que l'on propose de résoudre de façon gloutonne (voir le Tableau 3.4) en essayant d'insérer séquentiellement les sommets par ordre de poids $1 - \pi_v$ non croissants. Si l'ensemble θ-indépendant a un poids strictement supérieur à γ, alors il s'agit d'une colonne de coût réduit positif, que l'on peut transmettre au PMR.

Tableau 3.4 Une heuristique gloutonne pour le sous-problème

Algorithme 6: Heuristique de recherche d'un ensemble S_θ de poids maximum

```
1  INITIALISATION :
2  Ordonner les sommets par ordre non croissant de 1 − π_v dans une liste (v_1, ··· , v_n) ;
3  S_θ = ∅ ;
4  SÉLECTION :
5  pour v = v_1, ··· , v_n faire
6      si 1 − π_v > 0 alors
7          si S_θ ∪ {v} est θ-indépendant alors
8              S_θ ← S_θ ∪ {v} ;
9  SOLUTION :
10 si Σ_{v∈S_θ}(1 − π_v) > γ alors
11     retourner S_θ
```

Proposition 11. *Cette heuristique a une complexité algorithmique en $\mathcal{O}(n^2)$.*

Démonstration. Le tri initial se fait en $\mathcal{O}(n \log n)$. Ensuite, pour chaque sommet, il faut vérifier que son insertion dans S_θ ne crée pas de conflit ni pour lui, ni pour les membres de S_θ. On initialise $I_{S_\theta}(v) = 0$ pour tout sommet v. Lorsque v est inséré dans S_θ, on lui ajoute la quantité $\sum_{u \in S_\theta} \omega(u, v)$, et on ajoute $\omega(v, u)$ aux sommets u de S_θ, ce qui prend un temps

$\mathcal{O}(n)$. Ainsi, vérifier qu'un sommet est candidat à être inséré se fait aussi en temps linéaire, il suffit de vérifier que $I_{S_\theta}(v) + \sum_{u \in S_\theta} \omega(u, v) \leq \theta \cdot W(v)$ et que $I_{S_\theta}(u) + \omega(v, u) \leq \theta \cdot W(u)$ pour tout sommet u de S_θ. Puisque tous les sommets sont potentiellement candidats, l'algorithme a bien une compexité en $\mathcal{O}(n^2)$.

□

Coupes sur le problème maître

Il est tout à fait possible d'ajouter au problème maître les inégalités valides d'asymétrie (3.22)-(3.23) et d'incompatibilité (3.24), ainsi que la borne supérieure (3.21) basée sur la fonction de Lovász. Ces contraintes agissent comme des contraintes liantes supplémentaires.

Stratégies de branchement

Le processus de génération de colonnes est intégré dans un algorithme de branch-and-bound, par conséquent une fois la borne $\bar{z}$ trouvée, il faut définir des règles de branchement au cas où la solution courante est fractionnaire. Nous avons décidé de brancher sur les variables $x_{vi} = \sum_{p \in \Omega_i | v \in p} \lambda_{ip}$, ce qui revient à imposer $x_{vi} \geq \left\lceil \sum_{p \in \Omega_i | v \in p} \lambda_{ip} \right\rceil$ dans une branche et $x_{vi} \leq \left\lfloor \sum_{p \in \Omega_i | v \in p} \lambda_{ip} \right\rfloor$ dans l'autre si $x_{vi} \notin \{0, 1\}$. Cette stratégie de branchement est la stratégie par défaut et s'avère suffisante si la formulation III est très serrée. Nous verrons dans les tests numériques que bien souvent, c'est effectivement le cas.

Il est néanmois intéressant de mentionner quelques stratégies supplémentaires qui pourraient être envisagées. Tel que décrit dans [3], étant donnée une paire de sommets $\{u, v\}$, il pourrait s'avérer judicieux d'imposer qu'elle soit colorée de la même couleur dans une branche, et d'une couleur différente dans l'autre. Ceci peut être réalisé en ajoutant la contrainte $x_u + x_v \leq 1$ dans les sous-problèmes de la première branche, et $x_u = x_v$ dans ceux de l'autre. Le choix de la paire $\{u, v\}$ peut s'effectuer en sélectionnant celle dont la quantité $\sum_{p \in \Omega | \{u,v\} \subseteq p} \hat{\lambda}_p$ est la plus fractionnaire.

3.5 Résultats numériques

Choix des instances

On rappelle que les graphes utilisés sont complets. Les instances dépendent donc de $n = |V|$, du nombre de couleurs k, du paramètre θ, et des pondérations $W : V \to \mathbb{R}^+$, $\omega : V \times V \to \mathbb{R}^+$. Nous avons choisi des valeurs de n allant de 10 à 35, des valeurs de k allant de 1 à 9, et des valeurs de θ appartenant à l'ensemble $\{\frac{1}{4}, \frac{1}{2}, 1\}$ (rappelons que $\frac{1}{\theta}$ représente le SINR d'un

signal émis, et que la technologie actuelle impose typiquement $\frac{1}{\theta} < 2$). Les poids $W(v)$ et $\omega(u, v)$ sont des variables aléatoires uniformes sur l'intervalle $[1, n-1]$, l'idée étant d'avoir des graphes dont les sommets ont un degré entrant au minimum égal à 1. Ces instances aléatoires ne prétendent aucunement représenter un réseau de téléphonie mobile. Nous verrons dans le Chapitre 4 comment générer des instances plus fidèles aux réseaux sans fils, le but est ici de simplement valider les stratégies de résolution décrites ci-dessus.

Logiciel

Les tests numériques ont été effectués à l'aide de `DipPy` [98], un langage de modélisation basé sur Python, interfacé avec `DIP`, un solveur en open-source appartenant à la suite COIN-OR [1]. Tous les tests ont été effectués sur une machine MAC OS X, Intel Core i5, 1.80 GHz.

3.5.1 Comparaison de la solution optimale avec la borne de Lovász et les relaxations continues des formulations I, II, III

Dans un premier temps, on s'intéresse à la qualité des bornes supérieures obtenues avec la fonction généralisée de Lovász, notée $\overline{z}_L$, et en relâchant les contraintes d'intégralité sur les formulations I, II, III (z^I_{RL}, z^{II}_{RL}, z^{III}_{RL}). La solution optimale est notée z^*. Calculer la valeur exacte de la borne de Lovász n'est pas une tâche facile, cela requiert un algorithme de l'ellipsoïde [53]. Pour gagner en vitesse d'exécution, nous nous contenterons de calculer la somme des k plus grandes valeurs propres de la matrice d'adjacence, munie d'une diagonale de 1 (i.e. la restriction de $\vartheta_k(\bar{H})$ où $a_{ij} = 0$ si $i \neq j$ et $\{i, j\} \notin E$). Les résultats sont résumés dans le Tableau 3.5 ci-dessous, sur un total de 50 instances, avec θ fixé arbitrairement à 1.

Il est plus aisé d'analyser ces résultats à l'aide du graphe de la Figure 3.3, où l'on voit le pourcentage d'instances résolues avec une erreur relative inférieure à un certain seuil (par rapport à z^*), pour chaque valeur $\overline{z}_L$, z^I_{RL}, z^{II}_{RL}, z^{III}_{RL}. Par exemple, on observe que pour 90% des instances, la solution de la relaxation continue de la formulation III (z^{III}_{RL}) est optimale, et que pour toutes les instances, l'erreur relative est inférieure à 10%. Par contre, z^I_{RL} et z^{II}_{RL} ne sont jamais optimales ; pour environ 15% des instances l'erreur relative est inférieure à 20%, pour légèrement plus de 40% d'entre elles, elle est inférieure à 80%. Ceci confirme les résultats théoriques de la section 3.1, où l'on a vu que le polyèdre des contraintes de la formulation III est plus proche de son enveloppe convexe que les formulation I et II. On peut remarquer qu'il n'y a quasiment pas de différence entre la formulation I et II : la courbe associée à z^I_{RL} est en deçà de celle de z^{II}_{RL} sur uniquement un point. En ce qui concerne la borne généralisée de

1. Computational Infrastructure for Operations Research, `http://www.coin-or.org`

Lovász, elle est égale à z^* pour 10% des instances et pour environ 25% d'entre elles, l'erreur relative est inférieure à 20%. De manière générale, la courbe associée à $\overline{z}_L$ est au dessus de celles de z^I_{RL} et z^{II}_{RL}, ce qui veut dire que pour toute instance parmi les 50 utilisées, cette borne supérieure est plus serrée que celles obtenues en relâchant les contraintes d'intégralité des formulation I et II. On peut donc s'attendre à ce que l'utilisation de cette borne dans un algorithme de branch-and-bound diminue le nombre de nœuds de branchement, et par conséquent les temps d'exécution. Voyons ce qu'il en est.

Tableau 3.5 Comparaison des différentes bornes supérieures avec la solution optimale

(n,k)	$(10,1)$	$(10,2)$	$(10,3)$	$(10,4)$	$(10,5)$	$(15,1)$	$(15,2)$	$(15,3)$	$(15,4)$	$(15,5)$	$(15,6)$	$(15,7)$	$(15,8)$	$(15,9)$
z^*	3.0	5.0	7.0	8.0	9.0	3.0	5.0	7.0	9.0	10.0	11.0	12.0	13.0	14.0
$\overline{z}_L$	5.0	8.0	9.0	9.0	9.0	4.0	9.0	12.0	13.0	13.0	13.0	14.0	14.0	14.0
z^I_{RL}	5.4	10.0	10.0	10.0	10.0	7.7	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0
z^{II}_{RL}	5.3	10.0	10.0	10.0	10.0	7.7	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0
z^{III}_{RL}	3.0	5.0	7.0	8.5	9.5	3.0	5.0	7.0	9.0	10.0	11.0	12.0	13.0	14.0

(n,k)	$(20,1)$	$(20,2)$	$(20,3)$	$(20,4)$	$(20,5)$	$(20,6)$	$(20,7)$	$(20,8)$	$(20,9)$	$(25,1)$	$(25,2)$	$(25,3)$	$(25,4)$	$(25,5)$
z^*	3.0	6.0	8.0	10.0	12.0	14.0	16.0	18.0	19.0	3.0	6.0	9.0	12.0	14.0
$\overline{z}_L$	10.0	15.0	16.0	17.0	17.0	18.0	18.0	18.0	19.0	9.0	20.0	21.0	22.0	23.0
z^I_{RL}	10.3	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	12.8	25.0	25.0	25.0	25.0
z^{II}_{RL}	10.3	20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0	12.8	25.0	25.0	25.0	25.0
z^{III}_{RL}	3.0	6.0	8.0	10.0	12.0	14.0	16.0	18.0	19.0	3.0	6.0	9.0	12.0	14.0

(n,k)	$(25,6)$	$(25,7)$	$(25,8)$	$(25,9)$	$(30,1)$	$(30,2)$	$(30,3)$	$(30,4)$	$(30,5)$	$(35,1)$	$(35,2)$	$(35,3)$	$(35,4)$	$(35,5)$
z^*	16.0	18.0	20.0	22.0	3.0	6.0	9.0	11.0	13.0	3.0	6.0	9.0	12.0	14.0
$\overline{z}_L$	23.0	24.0	24.0	24.0	9.0	18.0	21.0	23.0	26.0	11.0	23.0	28.0	29.0	31.0
z^I_{RL}	25.0	25.0	25.0	25.0	15.2	30.0	30.0	30.0	30.0	17.7	35.0	35.0	35.0	35.0
z^{II}_{RL}	25.0	25.0	25.0	25.0	15.2	30.0	30.0	30.0	30.0	17.7	35.0	35.0	35.0	35.0
z^{III}_{RL}	16.0	18.0	20.0	22.0	3.0	6.0	9.0	11.5	13.5	3.0	6.0	9.0	12.0	14.0

3.5.2 Comparaison des différentes stratégies de résolution

Les Tableaux 3.6 et 3.7 présentent les résultats (nombre de nœuds de branchement et temps CPU en secondes) pour un nouvel ensemble d'instances, résolues à l'aide des stratégies suivantes :

1. Branch-and-bound :
   - Options par défaut de `DipPy` (-) ; [2],
   - Borne supérieure de Lovász (LUB) ;
   - LUB et contraintes d'asymétrie (OC) et d'incompatibilité (IC) ;

2. Dans ce cas, le solveur Dip résout le problème avec sa propre librairie de coupes, CGL (`http://www.coin-or.org/projects/Cgl.xml`) .

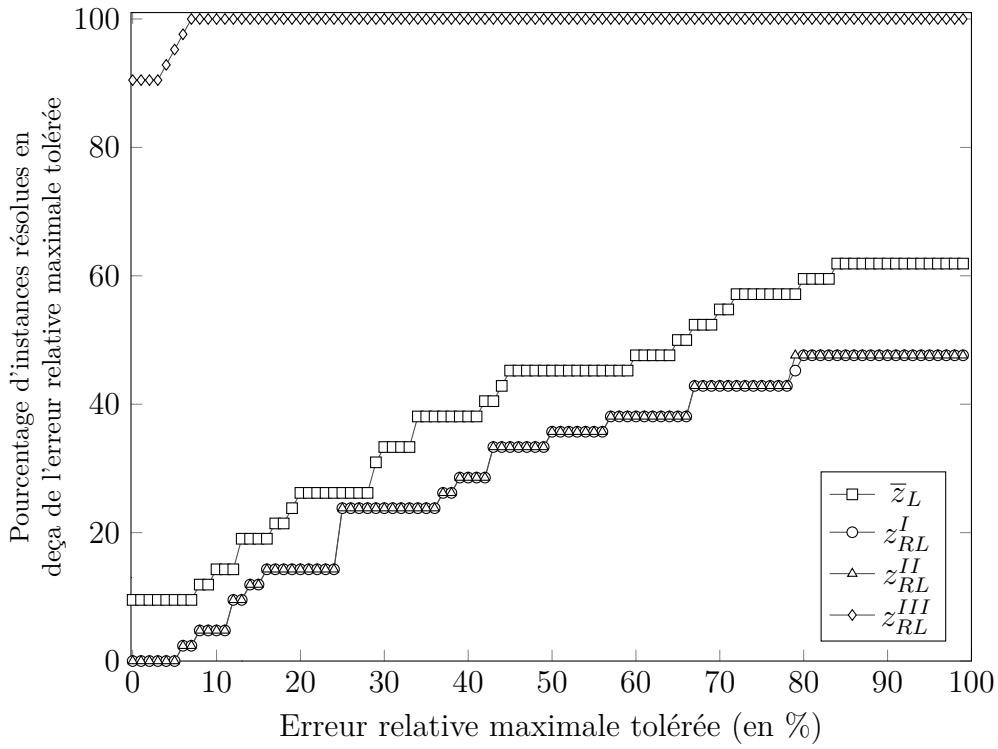


Figure 3.3 Graphe associé au Tableau 3.5

2. Branch-and-cut :
   - LUB, OC, IC, et coupes de recouvrement (CC) ;
   - LUB, OC, IC, et coupes de poids (WC) ;
   - LUB, OC, IC, et coupes de cliques (CLC) ;
   - LUB, OC, IC, et coupes de cycle impair (ODC) ;
3. Branch-and-price :
   - Génération de colonnes (CG) ;
   - CG et heuristique pour le sous-problème (SUB) ;
   - CG, SUB, OC, IC.

Il y a toujours un compromis entre le temps d'exécution et l'ajout de coupes. En effet, plus on ajoute de coupes, plus il est coûteux en temps de calcul de traiter un nœud de branchement, mais plus cela limite le nombre total de branches inutiles à explorer. Les solveurs permettent donc généralement de choisir la cadence avec laquelle on ajoute des coupes : de façon conservative, modérée, ou agressive. Dans notre cas, chaque type de coupe est valable pour les k couleurs, nous avons observé empiriquement que le choix d'une couleur prise au hasard donnait un bon compromis.

Le Tableau 3.6 présente les résultats numériques sur un graphe à $n = 10$ sommets, avec $k = 2, 3, 4, 5$ couleurs, et $\theta = 1$. On observe que l'ajout de la borne de Lovász diminue effectivement le nombre de nœuds de branchement dans la plupart des cas, mais de façon plus significative, c'est en la couplant avec les contraintes d'asymétrie et d'incompatibilité que les temps de calculs sont les plus diminués. Par exemple, pour l'instance $(n, k) = (10, 5)$, le nombre de nœuds de branchement avec la stratégie par défaut s'élève à 20 177, alors que seulement 1 236 sont nécessaires avec la stratégie LUB, OC, IC, ce qui a pour effet de diviser le temps de résolution par plus de 15. On observe également que l'ajout de coupes à chaque itération tend à diminuer le nombre de nœuds encore plus : sur cette même instance, la stratégie LUB, OC, IC, CLC ne donne plus que 17 nœuds de branchement, et moins d'une seconde suffit pour résoudre l'instance. Cependant, ce n'est pas toujours le cas : avec $k = 2, 3, 4$, l'ajout de coupes augmente parfois le nombre de nœuds, mais les temps de résolution demeurent compétitifs.

Tableau 3.6 Nœuds de branchements et temps de calculs avec $(n, \theta) = (10, 1)$

Instance	$(n,k)=(10,2)$		$(n,k)=(10,3)$		$(n,k)=(10,4)$		$(n,k)=(10,5)$	
Stratégie	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps
-	500	0.9	1134	3.7	3465	12.6	20177	85.9
LUB	434	1.1	2027	4.7	3276	11.9	17911	72.1
LUB & OC & IC	159	0.3	183	0.9	228	0.9	1236	5.4
LUB & OC & IC & CC	160	0.7	258	1.7	249	2.6	1148	8.7
LUB & OC & IC & WC	148	1.0	238	1.9	309	3.4	1239	13.6
LUB & OC & IC & CLC	104	2.0	52	1.2	8	0.5	17	0.9
LUB & OC & IC & ODC	177	1.5	183	2.3	206	3.6	869	10.3
CG	1	0.4	1	1.4	1	3.0	1	8.0
CG & SUB	1	0.4	1	1.0	1	2.3	1	7.6
CG & SUB & OC & IC	1	0.1	1	0.5	1	0.8	1	1.7

Il s'avère aussi que la résolution du problème à l'aide d'un algorithme de branch-and-price, via la formulation III, est une stratégie plus efficace que la stratégie du solveur par défaut. Les temps d'exécution sont en effet très rapides (de l'ordre de la seconde dans le meilleur des cas), principalement car un seul nœud de branchement suffit. Bien que l'ajout de l'heuristique pour le sous-problème améliore légèrement les temps de calculs (moins d'une seconde de gagnée), ceci n'apparaît pas comme un élément déterminant pour accélerer la résolution. En revanche, l'ajout de contraintes couplantes supplémentaires (OC,IC) permet de diminuer les temps de calculs : plus de 6 secondes sont gagnées avec $k = 5$.

L'évolution des temps d'exécution est représentée à la Figure 3.5, pour les stratégies par défaut, LUB, la meilleure stratégie branch-and-cut, et la meilleure stratégie branch-and-price. On observe que les temps de résolution augmentent de façon exponentielle avec la stratégie par défaut : 0.9 secondes suffisent pour l'instance $(n, k) = (10, 2)$, mais 85.9 secondes sont requises pour $(n, k) = (10, 5)$. Malgré le fait que les temps de la stratégie LUB sont légèrement inférieurs, l'augmentation demeure exponentielle (72.1 secondes pour $(n, k) = (10, 5)$). En revanche, les meilleures stratégies de branch-and-price permettent de maintenir des temps d'exécution de l'ordre de la seconde pour $n = 10$.

Les mêmes conclusions restent valables pour les graphes à $n = 15$ sommets, $k = 2, 3, 4, 5$ couleurs, et $\theta = 1$ (voir le Tableau 3.7 et la Figure 3.5) : les temps d'exécution augmentent exponentiellement avec la stratégie par défaut, légèrement moins avec la borne de Lovász, et significativement moins à l'aide de coupes ou d'un procédé de génération de colonnes. On observe également que l'ajout de coupes diminue le nombre de nœuds de branchements, mais au prix de temps calculs supérieurs à la stratégie (LUB,OC,IC). Les coupes de clique (CLC) semblent être les coupes les plus efficaces, et la stratégie de génération de colonnes couplée aux contraintes d'asymétrie et d'incompatibilité semble être la stratégie gagnante.

Tableau 3.7 Nœuds de branchements et temps de calculs avec $(n, \theta) = (15, 1)$

Instance	$(n,k) = (15,2)$		$(n,k) = (15,3)$		$(n,k) = (15,4)$		$(n,k) = (15,5)$	
Stratégie	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps
-	327	3.4	7529	63.1	72636	878.6	-	-
LUB	408	3.3	7019	58.1	59658	725.5	-	-
LUB & OC & IC	205	1.8	1190	11.0	2956	32.1	33151	338.55
LUB & OC & IC & CC	268	3.4	1257	16.4	2823	63.0	28671	473.77
LUB & OC & IC & WC	101	4.3	950	21.6	2284	66.0	19566	396.34
LUB & OC & IC & CLC	37	9.4	456	73.7	1114	94.0	8194	534.84
LUB & OC & IC & ODC	182	4.5	943	42.3	4926	129.9	50091	1290.66
CG	1	11.4	1	16.1	1	33.4	88	788.22
CG & SUB	1	10.6	1	14.7	1	32.0	-	-
CG & SUB & OC & IC	1	3.2	2	10.7	1	11.5	63	295.29

Ces instances ont également été résolues avec $\theta = 0.5$ (voir le Tableau 3.8) et $\theta = 0.25$ (voir le Tableau 3.9). Les résultats sont intéressants : il s'avère que dans ces cas, la borne de Lovász est serrée, et par conséquent les temps de calculs chutent jusqu'à la seconde. Pour $(n, k) = (10, 5)$, le nombre de nœuds de branchements est réduit de 36 880 à 27 avec la stratégie LUB, et à 4 avec OC,IC,CLC. Dans ce cas de figure, le branch-and-price, bien que compétitif, est légèrement plus coûteux en temps de calculs, tel qu'observé à la Figure 3.5.

3.6 Conclusion

Cette section avait pour objet la résolution optimale du (k, W, ω, θ)-PGSI par programmation en nombres entiers. Trois formulations ont été proposées, analysées, et améliorées au moyen d'inégalités valides. Ces inégalités valides sont de natures différentes : les premières sont issues de considérations logiques et sont ajoutées préalablement à la résolution, les secondes sont basées sur les problèmes classiques de sac-à-dos et de recouvrement et sont ajoutées au fur et à mesure de l'exécution de l'algorithme de branchement afin d'éliminer la solution fractionnaire courante. Nous avons également mis en évidence une borne supérieure sur l'objectif basée sur la fonction généralisée de Lovász $\vartheta_k(G)$, calculable en temps polynomial. Les tests numériques effectués sur des instances aléatoires ont montré que ces éléments permettent de retarder l'explosion combinatoire inhérente à tout problème $\mathcal{NP}$-difficile. De plus, la flexibilité du logiciel utilisé, à savoir `DipPy`, a permis de pousser ces comparaisons avec un algorithme de branch-and-price, intégrant un processus de génération de colonnes dans un arbre de branchement. Il s'avère que cette méthode permet de gagner en vitesse de résolution par rapport à la stratégie par défaut du solveur, essentiellement car la formulation associée (la formulation III) est extrêmement serrée.

L'objectif principal était de mettre en œuvre une méthode efficace pour obtenir la solution optimale, de façon à avoir une référence en vue de développer des heuristiques pour ce problème. Le problème d'affectation de canaux réel comprend des centaines voire des milliers de sommets, ce qui est évidemment hors de portée pour les méthodes exactes décrites ici. Voyons donc maintenant comment le résoudre dans ce cas de figure.

Tableau 3.8 Nœuds de branchements et temps de calculs avec $(n, \theta) = (10, 0.5)$

Instance	$(n,k) = (10,2)$		$(n,k) = (10,3)$		$(n,k) = (10,4)$		$(n,k) = (10,5)$	
Stratégie	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps
-	98	0.3	996	2.3	6990	15.7	36880	109.3
LUB	23	0.1	26	0.1	34	0.2	27	0.1
LUB & OC & IC	20	0.0	41	0.2	8	0.1	28	0.2
LUB & OC & IC & CC	20	0.1	41	0.3	8	0.1	28	0.4
LUB & OC & IC & WC	20	0.1	41	0.3	8	0.1	28	0.4
LUB & OC & IC & CLC	6	0.3	7	0.4	8	0.6	4	0.6
LUB & OC & IC & ODC	18	0.3	13	0.3	8	0.2	28	0.6
CG	1	0.1	1	0.6	1	1.0	1	1.5
CG & SUB	1	0.1	1	0.5	1	0.9	1	1.4
CG & SUB & OC & IC	1	0.1	1	0.2	4	1.0	1	0.6

Tableau 3.9 Nœuds de branchements et temps de calculs avec $(n, \theta) = (15, 0.25)$

Instance	$(n,k) = (15,2)$		$(n,k) = (15,3)$		$(n,k) = (15,4)$		$(n,k) = (15,5)$	
Stratégie	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps
-	367	2.5	7916	35.8	-	-	-	-
LUB	4	0.0	10	0.1	22	0.1	29	0.2
LUB & OC & IC	12	0.1	4	0.0	18	0.3	22	0.7
LUB & OC & IC & CC	12	0.2	4	0.1	18	0.5	22	1.0
LUB & OC & IC & WC	8	0.1	4	0.0	18	0.5	22	1.0
LUB & OC & IC & CLC	3	2.3	2	1.6	2	3.2	10	7.6
LUB & OC & IC & ODC	12	0.4	4	0.1	18	1.0	22	1.5
CG	1	0.9	1	2.0	1	4.5	1	6.0
CG & SUB	1	0.9	1	2.0	1	4.6	1	6.2
CG & SUB & OC & IC	1	0.4	1	1.7	1	1.5	1	1.6

Instance	$(n,k) = (15,6)$		$(n,k) = (15,7)$		$(n,k) = (15,8)$		$(n,k) = (15,9)$	
Stratégie	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps	Nœuds	Temps
-	-	-	-	-	-	-	-	-
LUB	6	0.1	36	0.5	33	0.2	66	1.6
LUB & OC & IC	8	0.3	3	0.1	3	0.3	22	0.8
LUB & OC & IC & CC	8	0.4	3	0.2	3	0.4	22	1.2
LUB & OC & IC & WC	8	0.4	3	0.2	3	0.4	9	0.7
LUB & OC & IC & CLC	2	1.8	9	9.9	6	9.1	11	8.0
LUB & OC & IC & ODC	8	0.7	3	0.3	3	0.6	22	1.6
CG	1	8.3	1	11.1	1	14.1	1	17.2
CG & SUB	1	31.3	1	13.1	1	15.5	1	19.2
CG & SUB & OC & IC	1	5.5	1	3.3	1	3.6	1	6.2

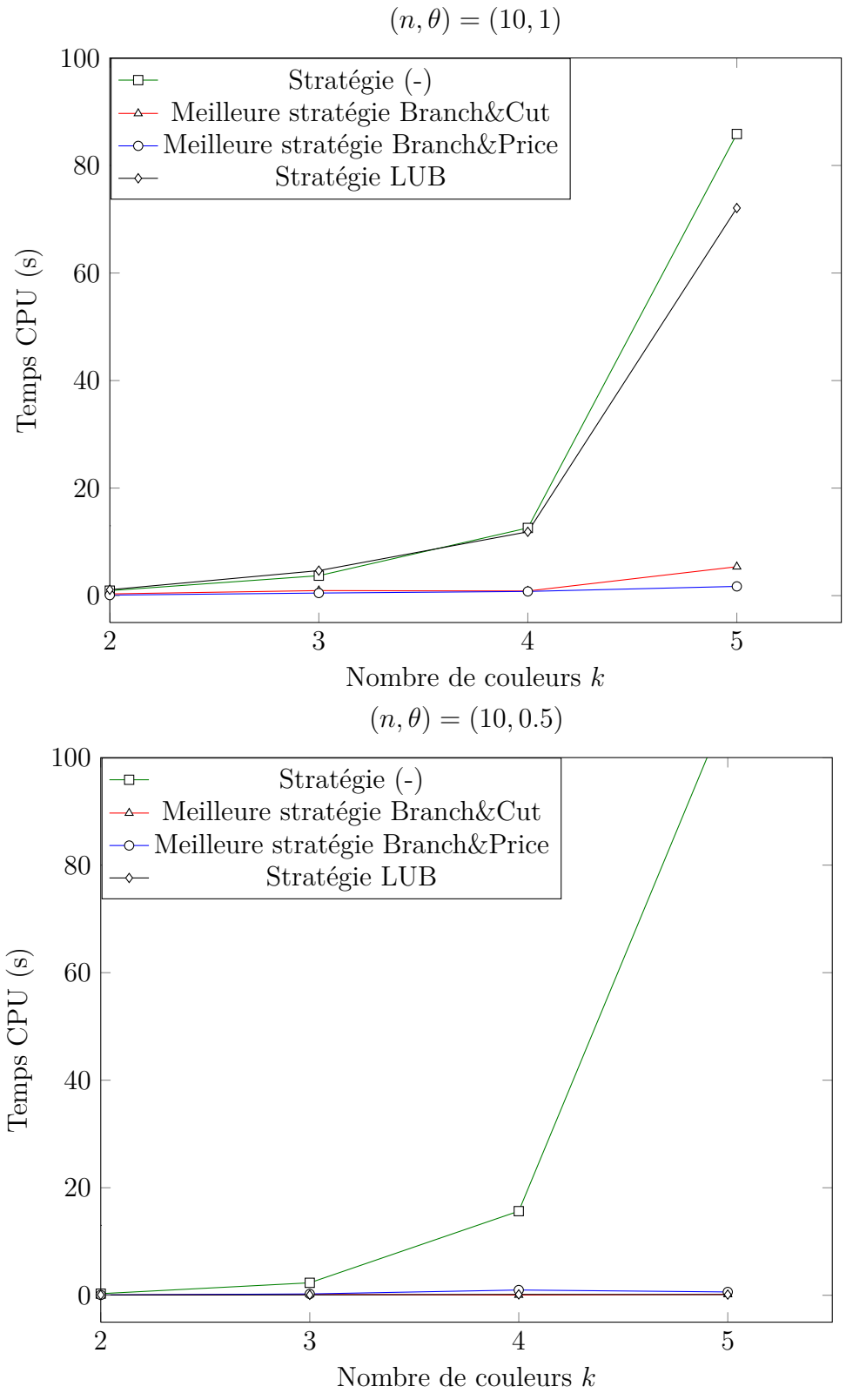


Figure 3.4 Évolution des temps de résolution, $n = 10$

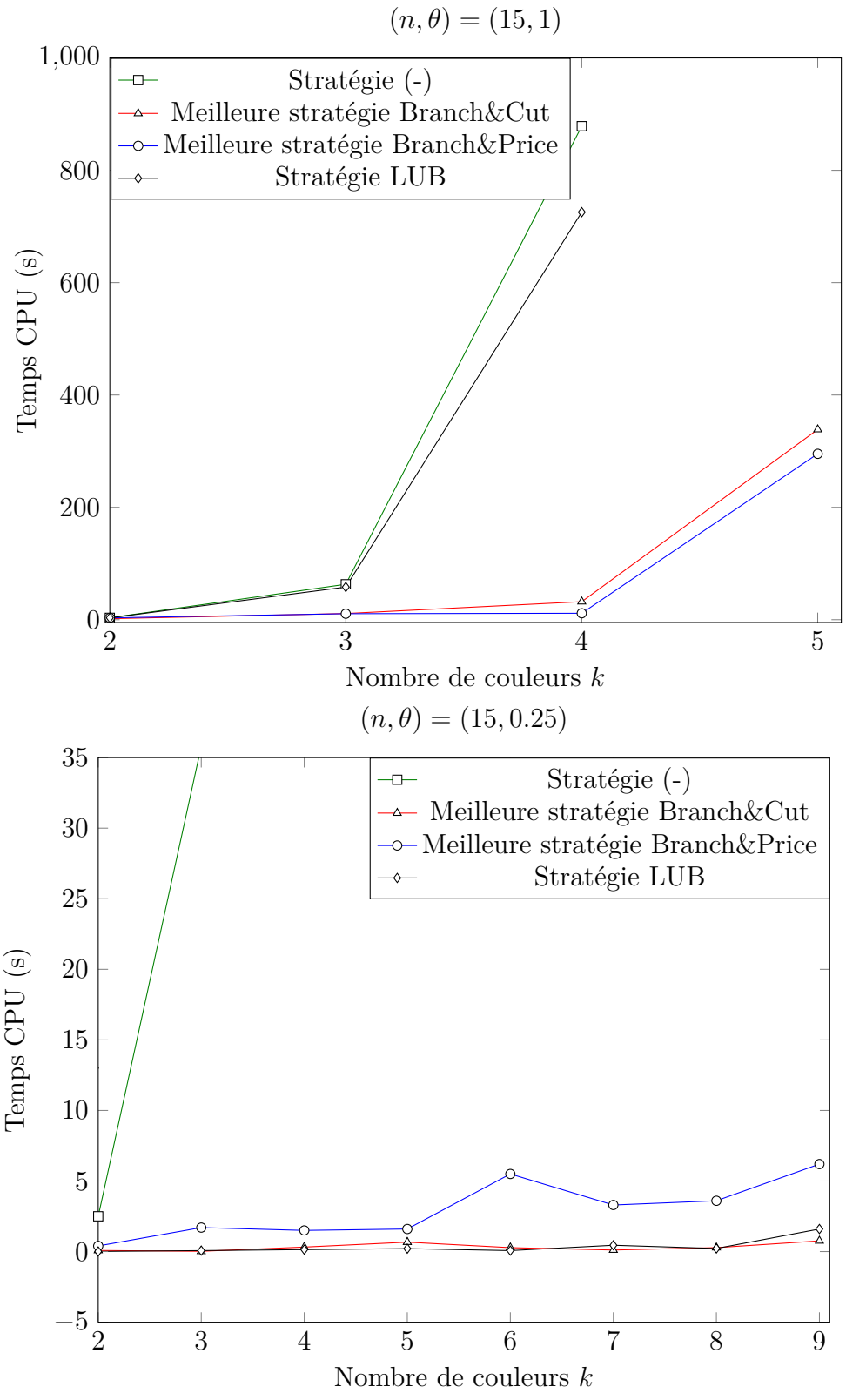


Figure 3.5 Évolution des temps de résolution, $n = 15$

CHAPITRE 4 RÉSOLUTION APPROCHÉE : HEURISTIQUES DE COLORATION CONSTRUCTIVES

Dans ce chapitre, nous allons voir comment adapter au (k, W, ω, θ)-PGSI les heuristiques LF, SLF et RLF présentées au Chapitre 2 . Le choix de ces trois algorithmes n'est pas complètement arbitraire : nous avons choisi deux algorithmes parmi ceux qui affectent séquentiellement une couleur aux sommets, et un parmi ceux qui construisent successivement des stables. Parmi ceux de la première catégorie, nous en avons adapté un où l'ordre des sommets est statique (LF), et un autre où il est dynamique (SLF). Les performances numériques de RLF évoquées plus haut ont motivé sa sélection au détriment de GIS, bien que celui-ci ait la meilleure garantie (pour le paramètre $\chi(G)$).

Ces heuristiques ont été développées en vue d'approcher le nombre chromatique $\chi(G)$, alors que c'est le paramètre $\alpha_k^\theta(G, W, \omega)$ que l'on vise, qui est une généralisation de $\alpha_k(G)$, le nombre maximum de sommets que l'on peut colorer avec k couleurs. Mais puisque l'on peut définir le nombre chromatique par

$$\chi(G) = \min\{k \in \mathbb{N} \mid \alpha_k(G) = n\},$$

ces deux problèmes d'optimisation peuvent être vus comme duaux l'un par rapport à l'autre ; exploiter les stratégies de LF, SLF ou RLF n'est alors pas dénué de sens dès lors que l'on fait l'hypothèse que les graphes sont entièrement colorables. Ceci est tout à fait raisonnable, car les réseaux de téléphonie mobile sont censés pouvoir assurer une couverture quasi-totale.

4.1 Préliminaires

L'algorithme LF affecte séquentiellement la plus petite couleur possible aux sommets ordonnés par ordre de degré non croissant. Il est possible de généraliser cette notion pour les graphes orientés, complets et pondérés : le lien entre un sommet et ses voisins peut être mesuré en évaluant la quantité $\deg(v, \omega)^- = \sum_{u \neq v} \omega(u, v)$. Nous allons voir comment poursuivre cette généralisation lorsque les sommets sont également pondérés. L'algorithme RLF exploite également cette notion de degré, mais de façon plus fine, en ne considérant qu'un sous-ensemble des sommets voisins. Ceci motive la définition suivante.

Définition 25. *Soit un graphe complet, orienté, pondéré sur les arcs par une fonction ω et sur les sommets par une fonction W. Le degré entrant pondéré d'un sommet v par rapport à un ensemble X donné est défini par :*

$$\deg_X^-(v, W, \omega) = \sum_{u \in X} \frac{\omega(u, v)}{W(v)}.$$

Si les pondérations W et ω sont constantes et égales, on retombe bien sur la cardinalité de l'ensemble X.

L'algorithme SLF ordonne dynamiquement les sommets par ordre de degré de saturation non croissant, où le degré de saturation d'un sommet v correspond au nombre de couleurs déjà utilisées par ses voisins :

$$\mathrm{dsat}(v) = \sum_{i=1}^{k} \min\{|V_i \cap N_v|, 1\},$$

$V_i \subseteq V$ étant l'ensemble des sommets de couleur i. Mais puisque les graphes étudiés ici sont orientés, complets, pondérés, et que la coloration est θ-impropre, cette notion doit être redéfinie, la notion de voisin n'ayant plus le sens commun du terme.

Définition 26. *Une couleur i est dite disponible pour un sommet non coloré v si les contraintes suivantes sont satisfaites :*

$$\sum_{u \in V_i} \omega(u, v) \leq \theta\, W(v) \tag{4.1}$$

$$\omega(v, u) + \sum_{x \in V_i | x \neq u} \omega(x, u) \leq \theta\, W(u) \qquad \forall u \in V_i. \tag{4.2}$$

La contrainte (4.1) assure que l'affectation de la couleur i à v ne créera pas de conflit pour v, et les contraintes (4.2) évitent les conflits pour les sommets u appartenant déjà à la classe de couleur i.

Définition 27. *Le degré de saturation* $\mathrm{dsat}(v)$ *d'un sommet v est alors le nombre de couleurs non disponibles pour v :*

$$\mathrm{dsat}(v, W, \omega) = |\{i \in \{1, \cdots, k\} \text{ tel que (4.1) ou (4.2) est violée}\}|.$$

Degré entrant pondéré, degré de saturation généralisé et couleur disponible sont les ingrédients principaux des heuristiques proposées ici. Voyons cela dans le détail.

4.2 Adaptation des algorithmes LF, SLF, RLF

Pour adapter l'algorithme LF, il suffit d'affecter la plus petite couleur possible en considérant les sommets par ordre de degré entrant pondéré non croissant. Si aucune couleur ne convient parmi l'ensemble $\{1, \cdots, k\}$, le sommet en question n'est pas coloré. Cela donne priorité aux sommets dont les liens avec ses prédécesseurs sont « forts », et donc difficiles à colorer. Cet algorithme, baptisté WP1 [1], est résumé dans le Tableau 4.1 ci-dessous.

Tableau 4.1 Adaptation de l'algorithme LF

Algorithme 7: WP1

```
1 INITIALISATION :
2 Ordonner les sommets v ∈ V par ordre non croissant de deg⁻_V(v, W, ω) dans une liste
  (v_1, ··· , v_n) ;
3 V_i = ∅ pour i = 1, ··· , k ;
4 COLORATION :
5 pour i = 1, ··· , k faire
6     pour v = v_1, ··· , v_n faire
7         si v n'est pas coloré et si la couleur i est disponible pour v alors
8             V_i ← V_i ∪ {v} ;
```

Pour SLF, on considère l'ensemble $A(v)$ des couleurs disponibles pour un sommet non coloré v (autrement dit $|A(v)| = k - \mathrm{dsat}(v, W, \omega)$), et l'ensemble U des sommets non colorés tels que $A(v) \neq \varnothing$. Initialement, on a donc $U = V$ et $A(v) = \{1, \cdots, k\}$. À chaque itération, on sélectionne un sommet $v \in U$ dont le nombre de couleurs disponibles $|A(v)|$ est minimal (ou tel que le degré de saturation est maximal) ; les cas d'égalité sont départagés par le plus grand degré entrant pondéré vis-à-vis des sommets non colorés. Le sommet v est alors affecté de la plus petite couleur de l'ensemble $A(v)$. Pour terminer l'itération, on met à jour les ensembles U et $A(u)$ en supprimant de U tous les sommets u pour lesquels $A(u)$ devient vide (ainsi que v, bien sûr), et l'on diminue les quantités $\deg^-_U(x, W, \omega)$ en conséquence pour les sommets $x \in U$ restants. Ceci permet de colorer en priorité les sommets les plus « urgents ». Cet algorithme, appelé DSAT1, est résumé dans le Tableau 4.2.

1. Cette abréviation est issue de Welsh & Powell, qui sont à l'origine de l'algorithme LF (voir le Chapitre 2).

Tableau 4.2 Adaptation de l'algorithme SLF

Algorithme 8: DSAT1

INITIALISATION :
$U = V$;
$\deg^-_U(v, W, \omega) = \deg^-_V(v, W, \omega)$ pour tout $v \in V$;
$V_i = \varnothing$ pour $i = 1, \cdots, k$;
$A(v) = \{1, \cdots, k\}$ pour tout $v \in V$;
COLORATION :
tant que $U \neq \varnothing$ **faire**
    Déterminer un sommet $v \in U$ de valeur minimale $|A(v)|$. En cas d'égalité, prendre celui qui maximise $\deg^-_U(v, W, \omega)$;
    Choisir la plus petite couleur $i \in A(v)$ et transférer v de U à V_i ;
    Mettre à jour les ensembles $A(u)$ pour tout $u \in U$ tels que $i \in A(u)$, et enlever de U les sommets pour lesquels $A(u) = \varnothing$;

Enfin, RLF construit une séquence $V_1, \cdots, V_k$ de classes de couleurs. Pour une itération donnée, par exemple la construction de V_i, appelons U l'ensemble des sommets non colorés pour lesquels la couleur i est disponible, et Z l'ensemble des sommets non colorés pour lesquels elle ne l'est plus. Ainsi, dès qu'un sommet est transféré de U à V_i, tous les sommets pour lesquels la couleur i n'est plus disponible sont transférés de U à Z. Le premier sommet $v \in U$ à être inséré dans V_i est celui de plus grande valeur $\deg^-_U(v, W, \omega)$, autrement dit celui qui semble a priori le plus difficile à colorer. Ensuite, tant que U n'est pas vide, on colore le sommet de plus grande valeur $\deg^-_Z(v, W, \omega)$, c'est en quelque sorte celui pour lequel un maximum de voisins sont déjà condamnés. Les cas d'égalité sont départagés par la plus petite valeur $\deg^-_U(v, W, \omega)$, autrement dit par le sommet qui va condamner le moins de sommets possibles. Bien sûr, les degrés $\deg^-_U(v, W, \omega)$ et $\deg^-_Z(v, W, \omega)$ sont mis à jour au fur et à mesure des transferts. Cet algorithme, appelé RLF1, est résumé dans le Tableau 4.3.

Tableau 4.3 Adaptation de l'algorithme RLF

Algorithme 9: RLF1

1 INITIALISATION :
2 $V_i = \varnothing$ pour $i = 1, \cdots, k$;
3 COLORATION :
4 **pour** $i = 1, \cdots, k$ **faire**
5 Soit U l'ensemble des sommets non colorés ; calculer $\deg_U^-(u, W, \omega)$ pour tout $u \in U$; Poser $Z = \varnothing$ et $\deg_Z^-(u, W, \omega) = 0$ pour tout $u \in U$;
6 Choisir un sommet $v \in U$ qui maximise $\deg_U^-(v, W, \omega)$;
7 Insérer v dans V_i et déplacer dans Z tous les sommets de U pour lesquels i n'est plus disponible ;
8 **tant que** $U \neq \varnothing$ **faire**
9 Choisir un sommet $u \in U$ qui maximise $\deg_Z^-(u, W, \omega)$; en cas d'égalité, choisir celui qui minimise $\deg_U^-(u, W, \omega)$;
10 Insérer u dans V_i, et déplacer dans Z tous les sommets de U pour lesquels i n'est plus disponible ;

4.3 Performances théoriques

4.3.1 Complexités algorithmiques

Nous avons vu à la Section 2.6.1 que LF, SLF et RLF avaient des complexités algorithmiques respectivement égales à $\mathcal{O}(m+n)$, $\mathcal{O}\left((m+n)\log n\right)$ et $\mathcal{O}(mn)$. On peut donc se demander si tel est toujours le cas pour leurs adaptations introduites ci-dessus. Pour la coloration classique, lorsque l'on attribue une couleur à un sommet v, il faut parcourir ses $\deg(v)$ arêtes pour déterminer si elle convient ou pas, soit un total de $\sum_{v \in V} deg(v) \in \mathcal{O}(m)$ opérations élémentaires (uniquement pour la phase de coloration). Ici, d'une part les graphes sont complets, donc $m \in \mathcal{O}(n^2)$, d'autre part nous avons vu que la notion de couleur disponible est régie par les équations (4.1) et (4.2). Voyons comment cela affecte les complexités de chacune des heuristiques.

Étant donnée une coloration de $G(V, W, \omega)$ en cours, on considère les quantités $I_\ell(v)$, définies pour tout sommet v, pour toute couleur ℓ par

$$I_\ell(v) = \sum_{u \in V_\ell, u \neq v} \omega(u, v).$$

Lorsqu'aucun sommet n'est coloré, $I_\ell(v) = 0$ pour tout sommet $v \in V$ et toute couleur $\ell \in \{1 \cdots, k\}$. Lorsqu'on attribue la couleur ℓ à un sommet v, les quantités $I_\ell(u)$ sont mises à jour en $\mathcal{O}(n)$ en posant

$$I_\ell(u) \leftarrow I_\ell(u) + \omega(v,u) \quad \forall u \in V, u \neq v.$$

De plus, pour un sommet non coloré v, $\mathcal{O}(|V_\ell|)$ opérations élémentaires sont nécessaires pour déterminer si la couleur ℓ est disponible, puisqu'il il faut vérifier que

$$I_\ell(v) \leq \theta\, W(v), \qquad I_\ell(u) + \omega(v,u) \leq \theta\, W(u) \qquad \forall u \in V_\ell.$$

Ces initialisations et mises à jour se font donc en un temps $\mathcal{O}(n^2)$, et l'on peut en déduire les complexités suivantes.

Proposition 12. *L'algorithme WP1 a une complexité en $\mathcal{O}(n^2)$.*

Démonstration. Lors de l'initialisation, le calcul des degrés se fait en un temps $\mathcal{O}(n^2)$, le tri des sommets se fait en $\mathcal{O}(n \log n)$, et le test d'une couleur ℓ pour un sommet v se fait en $\mathcal{O}(|V_\ell|)$ d'après la discussion ci-dessus. Par conséquent la construction de V_ℓ s'obtient en $\mathcal{O}(n|V_\ell|)$, et il s'ensuit que la complexité totale est en $\mathcal{O}(n \sum_{\ell=1}^{k} |V_\ell|) \subseteq O(n^2)$. □

Proposition 13. *L'algorithme DSAT1 a une complexité en $\mathcal{O}(n^3)$.*

Démonstration. Lors de l'initialisation, le calcul des degrés se fait en un temps $\mathcal{O}(n^2)$, et l'initialisation des ensembles $A(v)$ en $\mathcal{O}(nk)$. À chaque fois qu'un sommet w est supprimé de U, les degrés $\deg^-_U(u, W, \omega)$ de tous les sommets $u \in U$ sont mis à jour en soustrayant la quantité $\frac{\omega(w,u)}{W(u)}$. Par conséquent, l'ensemble de ces mises à jour se fait en $\mathcal{O}(n^2)$. Le choix du sommet prioritaire et de sa couleur ℓ se fait en $\mathcal{O}(n + k)$, et les mises à jour de l'ensemble $A(u)$ pour les sommets $u \in U$ tels que $\ell \in A(u)$ nécessite de l'ordre de $\mathcal{O}(|V_\ell|)$ opérations élémentaires par sommet, soit un total de $\mathcal{O}(n|V_\ell|) \subseteq \mathcal{O}(n^2)$ opérations. Étant donné qu'il faut colorer n sommets, cela donne bien une complexité totale en $\mathcal{O}(n^3)$. □

Proposition 14. *L'algorithme RLF1 a une complexité en $\mathcal{O}(n^3)$.*

Démonstration. Considérons la construction d'une classe quelconque V_ℓ. Les valeurs des $\deg^-_U(u, W, \omega)$ pour les sommets $u \in U$ s'obtiennent en $\mathcal{O}(n^2)$, l'initialisation de l'ensemble Z, de $\deg^-_Z(u, W, \omega)$, et du choix du sommet v qui maximise $\deg^-_U(v, W, \omega)$ se font en $\mathcal{O}(n)$. Ensuite, les transferts de sommets de U vers Z s'obtiennent également en $\mathcal{O}(n)$, puisqu'il suffit de vérifier si $\omega(u, v) > \theta W(v)$ ou $\omega(v, u) > \theta W(u)$. Analysons maintenant le nombre

d'opérations requises pour mettre à jour $\deg_U^-(u, W, \omega)$ et $\deg_Z^-(u, W, \omega)$ pour les sommets $u \in U$ lorsqu'ils sont transférés de U vers Z ou V_ℓ. À chaque fois qu'un sommet w est transféré de U vers Z, $\deg_Z^-(x, W, \omega)$ est incrémenté de $\frac{\omega(w,x)}{W(x)}$, et $\deg_U^-(x, W, \omega)$ est décrémenté de $\frac{\omega(w,x)}{W(x)}$, et ceci pour tout $x \in U$. De plus, lorsqu'un sommet $v \in U$ est transféré de U vers V_ℓ, $\deg_U^-(x, W, \omega)$ diminue de $\frac{\omega(v,x)}{W(x)}$ pour tout $x \in U$. Puisque tous les sommets de U sont transférés soit vers Z, soit vers V_ℓ, cela donne un total de $\mathcal{O}(n^2)$ mises à jour, toutes effectuées en temps constant. Lorsque la couleur ℓ est affectée à un sommet, il faut vérifier si elle reste disponible pour les sommets de U, ce qui se fait en $\mathcal{O}(|U||V_\ell|) \subseteq \mathcal{O}(n|V_\ell|)$, par conséquent de l'ordre de $\mathcal{O}(n|V_\ell|^2)$ opérations sont requises avant que $U = \varnothing$. Finalement, la construction d'une classe V_ℓ se fait en $\mathcal{O}(n^2 + n|V_\ell|^2)$, et puisque k classes sont construites successivement, il s'ensuit que l'algorithme a une complexité totale en $\mathcal{O}(kn^2 + n\sum_{\ell=1}^{k} |V_\ell|^2) \subseteq \mathcal{O}(n^3)$. □

Ainsi, les complexités des adaptations de LF et RLF demeurent inchangées (puisque $m \in \mathcal{O}(n^2)$, mais celle de SLF a légèrement régressé de $\mathcal{O}((m+n)\log n)$ à $\mathcal{O}(n^3)$.

4.3.2 Plus petits graphes sous-optimaux

Pour trouver les plus petits graphes sous-optimaux de ces algorithmes, nous allons revenir temporairement à la coloration classique (0-impropre, non orientée, sans aucune pondération) et considérer le paramètre $\alpha_k(G)$. Les algorithmes qui précèdent demeurent tels quels ; en posant $\theta = 0$, $W = \omega = 1$, et en remplaçant une arête $\{u, v\}$ par deux arcs (u, v) et (v, u), ils sont directement applicables à ce cas de figure. Nous verrons dans un deuxième temps pourquoi les résultats qui suivent restent valables pour la coloration θ-impropre pondérée.

Proposition 15. *Considérons l'algorithme Δ-Glouton qui sélectionne en priorité un sommet de plus grand degré parmi les non colorés, et qui lui attribue la plus petite couleur possible parmi l'ensemble $\{1, \cdots, k\}$. Alors, le plus petit graphe sous-optimal (par rapport au paramètre $\alpha_k(G)$) a au moins $k + 2$ sommets.*

Démonstration. Si $k = 1$, c'est trivial : sur un graphe à deux sommets, l'algorithme trouvera toujours la solution optimale, donc il faut au moins trois sommets. Par exemple, le chemin P_3 est sous-optimal (le sommet du milieu de degré 2 sera coloré, alors que le plus grand stable contient les deux sommets aux extrémités). Supposons donc que la propriété soit vraie au rang k, et soit G_k le plus petit graphe sous-optimal. Par hypothèse, G_k a donc au moins $k + 2$ sommets. Si l'on dispose d'une couleur supplémentaire, il est possible de constuire un graphe sous-optimal de taille $k + 3$ à partir de G_k : il suffit d'ajouter un sommet relié à tous les sommets de G_k. Ce sommet a le plus grand degré et sera coloré en premier, et puisqu'il est relié à tous les autres sommets, aucun autre ne pourra prendre cette couleur. De plus,

tous les sommets de G_k ont leur degré augmenté d'une unité, par conséquent G_k sera coloré comme précédemment avec les k couleurs restantes, c'est-à-dire de façon sous-optimale. Il s'ensuit que G_{k+1} a au moins $k+3$ sommets. □

Nous disposons donc d'une condition nécessaire pour obtenir un graphe sous-optimal par rapport au paramètre $\alpha_k(G)$ si l'on utilise Δ-Glouton. Il n'est pas difficile de créer un tel graphe : il suffit de considérer une clique K_{k+2} (c'est-à-dire un sous-graphe complet d'ordre $k+2$), à laquelle on retire l'arête entre deux sommets quelconques u et v. Appelons cette famille de graphes G_k.

Proposition 16. *Les graphes G_k sont sous-optimaux si l'on utilise l'algorithme Δ-Glouton.*

Démonstration. Par définition, Δ-Glouton va d'abord considérer tous les sommets de l'ensemble $K_{k+2} \setminus \{u, v\}$ de degré $k+1$, épuisant les k couleurs disponibles. Or, la solution optimale consiste à colorer tous les sommets de G_k, sauf un parmi les sommets de degré $k+1$. La Figure 4.1 ci-dessous illustre une coloration sous-optimale et une optimale pour G_1, G_2, G_3. □

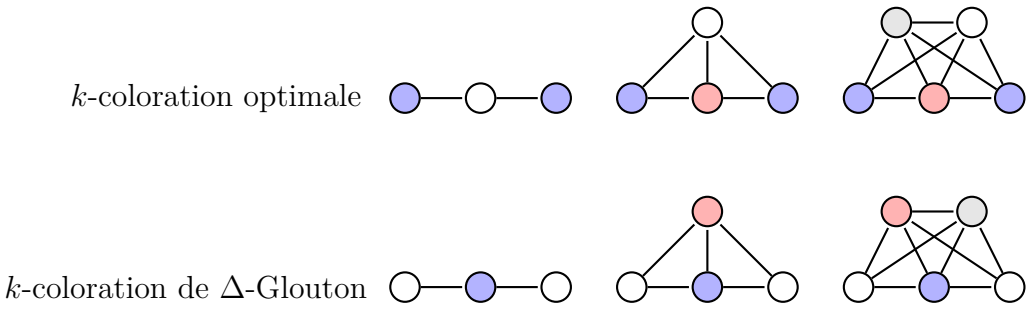


Figure 4.1 De gauche à droite : G_1, G_2, G_3

Il s'avère que les algorithmes WP1, DSAT1 et RLF1 sont tous Δ-Gloutons, d'où le corollaire suivant.

Corollaire 5. *Les graphes G_k sont les plus petits graphes sous-optimaux pour WP1, DSAT1 et RLF1.*

Démonstration. Les trois algorithmes, à un moment donné, considèrent les sommets de plus grand degré. C'est clair pour WP1. Pour DSAT1, le critère de priorité est le degré de saturation, mais les cas d'égalité sont départagés par le plus grand degré. Ainsi, le premier sommet coloré est un sommet de la clique, et le degré de saturation de tous les autres sommets augmente alors d'une unité. Le deuxième sommet est donc à nouveau un sommet de

la clique, et ainsi de suite. Pour RLF, le premier sommet intégré dans la première classe de couleur et celui de plus grand degré vis-à-vis des sommets non colorés, c'est-à-dire un parmi la clique. La classe est alors saturée, et le degré vis-à-vis des sommets non colorés diminue d'une unité pour tous les sommets non colorés. C'est donc à nouveau un sommet de la clique qui constitue la deuxième classe de couleur, et ainsi de suite. □

Retour au cas orienté, complet, pondéré, θ-impropre

Les résultats précédents sur les graphes classiques sont généralisables sur les graphes étudiés dans cette thèse, et ceci quelque soit le paramètre θ. Il suffit de remarquer que l'on peut toujours imposer que deux sommets u et v ne puissent pas être colorés avec la même couleur en créant un arc (u, v), pondéré par n'importe quelle valeur $\omega(u, v) > \theta W(v)$. Inversement, on peut toujours s'assurer qu'ils le puissent, en pondérant tous les arcs entrant vers v et u par n'importe quelle valeur inférieure à $\frac{\theta W(v)}{n-1}$, $\frac{\theta W(u)}{n-1}$, respectivement. Autrement dit, on peut toujours construire des cliques ou des ensembles indépendants, et donc conserver la structure des graphes G_k.

Contrairement à leur version initiale (pour la recherche de $\chi(G)$), les adaptations de LF, SLF et RLF ont donc toutes le même plus petit graphe sous-optimal, pour une valeur de k fixée.

4.3.3 Garanties de performance

Il a été prouvé [52] qu'aucun algorithme polynomial ne peut être ε-compétitif pour le problème du plus grand stable, pour tout $\varepsilon < 1$. Puisqu'il s'agit d'un cas particulier du plus grand sous-graphe induit k-colorable de façon θ-impropre (avec $W = \omega = k = 1$ et $\theta = 0$), on ne peut espérer faire mieux ici.

D'ailleurs, si l'on revient momentanément à la coloration classique, il n'est pas difficile de créer un graphe pour lequel l'écart entre la taille de la solution obtenue par Δ-Glouton et celle de la solution optimale est de l'ordre de $\mathcal{O}(n)$. En se basant sur les graphes G_k, il suffit d'ajouter p sommets indépendants reliés à tous les sommets d'une clique K_k (voir la Figure 4.2). Tous les sommets de K_k ont leur degré augmenté de p unités et seront à nouveau colorés en premier, alors que la solution optimale consiste à colorer tous les sommets du graphe, sauf un parmi K_k. L'ensemble des sommets indépendants pouvant être arbitrairement grand, il s'ensuit que Δ-Glouton ne pourra garantir mieux que $\mathcal{O}(\frac{k}{n})$.

On peut néanmoins aisément établir une garantie de performance en $\Delta(G)$, ce qui peut être intéressant pour les graphes dont le degré est borné.

Coloration de Δ-Glouton Coloration optimale

Figure 4.2 $\dfrac{\Delta\text{-Glouton}(G)}{\alpha_k(G)} = \dfrac{k}{n-1}$

Proposition 17. *Soit un graphe de degré maximum $\Delta(G)$. L'algorithme qui construit un k-stable de façon gloutonne (c'est-à-dire en attribuant la plus petite couleur possible à un sommet donné) garantit de trouver une solution comprenant au moins $\dfrac{\alpha_k(G)}{\Delta(G)+1}$ sommets.*

Démonstration. Par construction, l'algorithme génère un sous-graphe comprenant un ensemble de A sommets, maximal pour l'inclusion. Par conséquent, un sommet non coloré a au moins un voisin parmi l'ensemble des A sommets, et il s'ensuit que le nombre de sommets non colorés $n-$A est borné par $\Delta(G)$A. En isolant A, on obtient

$$A \geq \frac{n}{\Delta(G)+1} \geq \frac{\alpha_k(G)}{\Delta(G)+1}.$$

□

De même que leurs versions originales, les adaptations de LF, SLF, RLF ont donc la pire garantie théorique possible en $\mathcal{O}(\frac{k}{n})$.

4.4 Améliorations

Les simulations numériques présentées à la Section 4.6 montreront que WP1, DSAT1 et RLF1 ont tendance à affecter la même couleur à deux sommets u et v pour lesquels $\omega(u,v)$ ou $\omega(v,u)$ est proche de $\theta\, W(v)$ ou de $\theta\, W(u)$, respectivement. Ceci a pour effet de saturer rapidement la classe de couleur en question. Nous proposons deux façons d'éviter cela : la première exploite la structure des réseaux, plus particulièrement la répartition des stations de base et des usagers dans le plan, la seconde affine la notion de couleur disponible. Dans les deux cas nous allons proposer des règles de priorité supplémentaires.

4.4.1 Pavages de Voronoï

La première amélioration suppose que G soit construit à partir d'un ensemble $S = \{s_1, \cdots, s_t\}$ de stations de base, et d'un ensemble de n usagers, tel qu'expliqué au Chapitre 2. Ceci suppose que tous les sommets de G aient des coordonnées géographiques dans le plan. Rappelons aussi que la puissance $P_{va(v)}$ émise par un usager v à la station $a(v)$ est inversement proportionnelle à $d^{\gamma}_{va(v)}$, la distance entre v et la station $s_{a(v)}$ à une certaine puissance comprise entre 2 et 4 selon le milieu environnant. Comment peut-on exploiter ce phénomène ?

Si un sommet v donné est très proche de la station de base $a(v)$, son coefficient $W(v) = P_{va(v)}$ sera d'autant plus important, et v pourra donc supporter beaucoup de signaux interférents, de poids $\omega(u, v)$. Par conséquent v pourrait a priori prendre n'importe quel canal. Inversement, si son signal est faible à cause de son éloignement, ce sera un sommet vulnérable aux interférences, et il ne faudrait pas qu'il soit affecté du même canal qu'un sommet u proche géographiquement, dont le poids $\omega(u, v)$ est sensiblement égal à celui de $W(v)$. Pour éviter cela, il serait judicieux d'affecter en priorité des canaux différents à u et v. Voyons comment réaliser cela plus en détail.

La région de Voronoï C_p associée à la station de base s_p contient par définition tous les usagers plus proches de s_p que de n'importe quelle autre station. Idéalement, il faudrait que deux usagers u et v de faibles poids $W(u)$ et $W(v)$ et utilisant le même canal soient séparés par au moins une région de Voronoï, de façon à ce que la distance qui les sépare assure d'avoir des coefficients $\omega(u, v)$ et $\omega(v, u)$ également faibles. On va donc chercher à déterminer des sous-ensembles de canaux prioritaires pour chaque région de Voronoï, de sorte que ces sous-ensembles soient distincts pour deux régions ayant une frontière commune. On reconnaît un problème de coloration d'une carte géographique (tel que décrit au Chapitre 2), où une couleur associée à une région représente un sous-ensemble de canaux pouvant être affectés à un usager dans cette région. Commençons donc par déterminer une telle coloration. Idéalement, il faudrait aussi maximiser la distance entre deux régions de même couleur. Pour ce faire, il suffit de résoudre le programme en nombre entiers décrit dans le paragraphe qui suit.

Soit d_{pq} la distance Euclidienne entre s_p et s_q. Considérons le dual du pavage de Voronoï : le graphe planaire $H = (V_H, E_H)$ ayant comme ensemble de sommets $V_H = S$ (i.e. un sommet par station), et comme ensemble d'arêtes $E_H = \{(s_p, s_q) \mid C_p \text{ et } C_q \text{ sont adjacentes}\}$. Autrement dit, colorer le pavage est équivalent à colorer son graphe dual. Soit D une constante égale à $\max_{1 \leq p,q \leq t}\{d_{pq}\}$, soit δ une variable réelle positive représentant la distance minimale entre deux sommets de V_H de même couleur, soit x_{pi} une variable binaire égale à 1 si et seulement si s_p prend la couleur i, soit y_{pq} une variable binaire égale à 1 si et seulement si s_p et s_q reçoivent la même couleur, et soit z_i une autre variable binaire qui prend la valeur

1 si et seulement si la couleur i est utilisée. La coloration du pavage de Voronoï s'obtient en résolvant le modèle suivant :

$$\min_{x,y,z,\delta} \quad D\sum_{i=1}^{4} z_i - \delta \tag{4.3}$$

$$\text{sujet à} \quad \sum_{i=1}^{4} x_{pi} = 1 \qquad \forall s_p \in V_H \tag{4.4}$$

$$x_{pi} + x_{qi} \leq z_i \qquad \forall (s_p, s_q) \in E_H, \quad \forall i \in \{1, \cdots, 4\} \tag{4.5}$$

$$x_{pi} + x_{qi} \leq y_{pq} + 1 \qquad \forall s_p, s_q \in V_H, p \neq q, \quad \forall i \in \{1, \cdots, 4\} \tag{4.6}$$

$$\delta \leq d_{pq} + D(1 - y_{pq}) \qquad \forall (s_p, s_q) \notin E_H \tag{4.7}$$

$$x_{pi},\ y_{pq},\ z_i \in \{0, 1\} \qquad \forall s_p, s_q \in V_H, \quad \forall i \in \{1, \cdots, 4\} \tag{4.8}$$

$$\delta \geq 0 \tag{4.9}$$

Tel que mentionné au Chapitre 2, le théorème des quatre couleurs [45] stipule qu'un graphe planaire peut toujours être coloré en au plus 4 couleurs. On peut donc imposer dans les contraintes (4.4) que chaque sommet de H reçoive exactement une couleur parmi l'ensemble $\{1, \cdots, 4\}$. Les contraintes (4.5) assurent que deux sommets adjacents de H ne puissent pas avoir la même couleur. Elles permettent également d'activer le compteur de couleurs via les variables z_i. Les contraintes (4.6) lient les variables x et y, et les contraintes disjonctives (4.7) imposent que δ soit bornée par la distance minimum entre deux sommets de même couleur. Enfin, l'objectif (4.3) vise à trouver une coloration utilisant le moins de couleurs possible tout en maximisant la valeur de δ. Le fait d'introduire la constante D dans l'objectif permet de toujours favoriser l'utilisation de moins de quatre couleurs lorsque cela possible, même si cela dégrade la valeur de δ. En effet, δ étant borné par D, une solution à 4 couleurs donnera un objectif supérieur ou égal à $4D - D = 3D$, ce qui est bien supérieur à l'objectif obtenu avec trois couleurs : $3D - \delta$.

Ce modèle en nombres entiers présente l'avantage de maximiser la plus petite distance entre deux régions de même couleur, mais peut être difficile à résoudre si le nombre de stations est important. Cependant, le théorème des quatre couleurs fournit un algorithme en $O(n^2)$ permettant de trouver une solution réalisable à ce problème [103], il est donc toujours possible d'avoir une coloration des sommets de H. Soit $\chi \leq 4$ le nombre de couleurs utilisées, et soit $c(p) \in \{1, \cdots, \chi\}$ la couleur assignée au sommet s_p. En supposant que le nombre total de canaux k soit un multiple de 12, on associe alors à chaque région C_p un sous-ensemble de canaux F_p de sorte que $F_p \cap F_q = \varnothing$ si C_p et C_q ont une frontière commune, et tel que tous les sous-ensembles F_p contiennent $\frac{k}{\chi}$ canaux. Il suffit pour cela de poser

$$F_p = \left\{ \frac{(c(p)-1)k}{\chi} + 1, \cdots, \frac{c(p)k}{\chi} \right\} \quad \forall p \in \{1, \cdots, t\}.$$

Par exemple, si $k = 120$ canaux sont disponibles, $\chi = 3$ et $c(p) = 2$, alors $F_p = \{41, \cdots, 80\}$, tandis que si $k = 120$, $\chi = 4$ et $c(p) = 2$, alors $F_p = \{31, \cdots, 60\}$. Ces sous-ensembles de canaux permettent de définir des règles de priorité pour les usagers v dont le poids $W(v)$ est faible. Plus précisément, un usager v dans la cellule C_p recevra en priorité un canal parmi l'ensemble $S(v) = F_p$ s'il est éloigné de la station de base $a(v) = s_p$, mais s'il s'y trouve à proximité, il peut a priori prendre n'importe quel canal, et donc $S(v) = \{1, \cdots, k\}$. La coloration du pavage est telle que deux usagers utilisant le même canal seront soit séparés par au moins une région, soit à proximité d'une station de base et donc avec un poids suffisamment élevé. Il reste à déterminer la limite à partir de laquelle on considère que v passe de la zone centrale de C_p à une zone en bordure et devient candidat pour recevoir un canal prioritaire, ou en d'autres mots la limite à partir de laquelle on considère que son poids est trop faible. Cette limite est définie comme suit.

Définition 28. *Soit τ un paramètre réel appartenant à l'intervalle $[0, 1]$ et soient $a(v)$ et $b(v)$ les première et deuxième stations les plus proches de l'usager v, respectivement. Alors v est dans la zone centrale de la cellule $C_{a(v)}$ si et seulement si $\frac{P_{vb(v)}}{P_{va(v)}} \leq \tau$; sinon, il est en zone de bordure.*

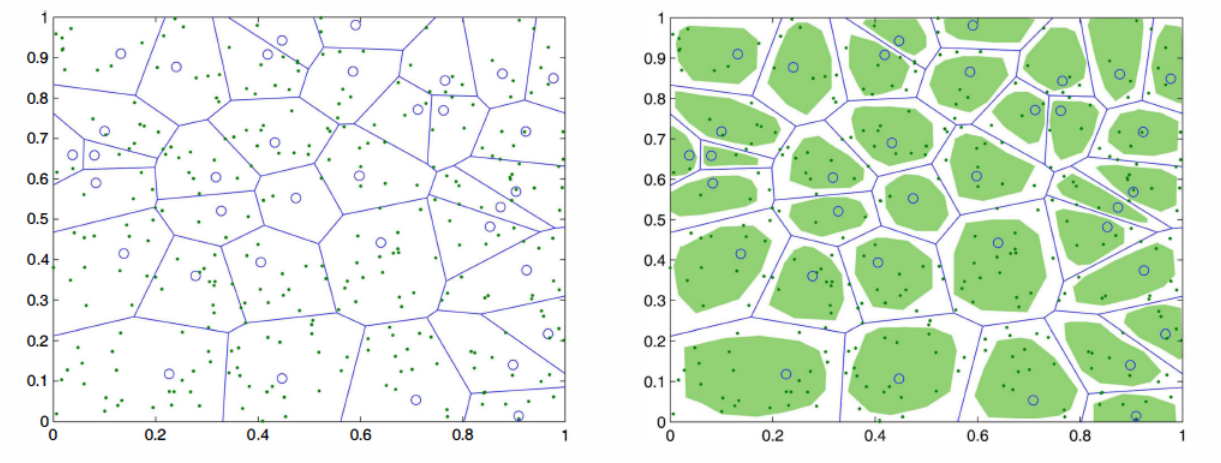


Figure 4.3 Pavage de Voronoï - zones centrale (en vert) et en bordure (en blanc)

Ainsi, si $\tau = 0$, tous les usagers sont en bordure de leur cellule s_p et sont candidats pour prendre en priorité une couleur parmi le sous-ensemble F_p. Inversement, si $\tau = 1$, ils sont tous considérés comme étant au centre, et aucune règle de priorité n'est appliquée. Un exemple

(emprunté à Rahat Ullah et al. [104]) est illustré à la Figure 4.3. Les stations de base sont représentées par des cercles bleus, et les usagers par des points verts ; les zones centrales des cellules sont colorées en vert ; tous les usagers se trouvant dans une zone blanche sont donc candidats pour prendre une couleur prioritaire. Des tests numériques montreront qu'aucune valeur de τ ne semble garantir une meilleure solution. Nous utiliserons donc 11 valeurs différentes $(0, 0.1, 0.2, \cdots, 1)$ et conserverons la meilleure solution obtenue.

Considérons dans ce qui suit que tout sommet v de G est affecté du sous-ensemble $S(v)$ de couleurs prioritaires. WP2, DSAT2 et RLF2 sont des versions modifiées de WP1, DSAT1 and RLF1, qui prennent en compte ces préférences en affectant une couleur $i \in S(v)$ à $v \in V$ lorsque cela possible. Ces versions sont résumées dans les Tableaux 4.4 - 4.6. En considérant que les ensembles $S(v)$ sont donnés, les complexités algorithmiques ne sont pas affectées.

Tableau 4.4 Deuxième adaptation de l'algorithme LF

Algorithme 10: WP2

INITIALISATION :
Ordonner les sommets $v \in V$ par ordre non croissant de $deg_V^-(v, W, \omega)$ dans une liste $(v_1, \cdots, v_n)$;
$V_i = \varnothing$ pour $i = 1, \cdots, k$;
pour $\tau = 0, 0.1, 0.2, \cdots, 1$ **faire**
    Déterminer les ensembles de couleurs prioritaires $S(v)$ pour $v \in V$;
    COLORATION :
    pour $i = 1, \cdots, k$ **faire**
        pour $v = v_1, \cdots, v_n$ **faire**
            si *v n'est pas coloré, $i \in S(v)$ et i est disponible pour v* **alors**
                $V_i \leftarrow V_i \cup \{v\}$;
        pour $v = v_1, \cdots, v_n$ **faire**
            si *v n'est pas coloré et si la couleur i est disponible pour v* **alors**
                $V_i \leftarrow V_i \cup \{v\}$;
retourner La meilleure solution obtenue ;

Tableau 4.5 Deuxième adaptation de l'algorithme SLF

Algorithme 11: DSAT2

INITIALISATION :
$U = V$;
$\deg^-_U(v, W, \omega) = \deg^-_V(v, W, \omega)$ pour tout $v \in V$;
$V_i = \varnothing$ pour $i = 1, \cdots, k$;
$A(v) = \{1, \cdots, k\}$ pour tout $v \in V$;
pour $\tau = 0, 0.1, 0.2, \cdots, 1$ **faire**
    Déterminer les ensembles de couleurs prioritaires $S(v)$ pour $v \in V$;
    COLORATION :
    tant que $U \neq \emptyset$ **faire**
        Déterminer un sommet $v \in U$ de valeur maximale $|A(v)|$. En cas d'égalité, prendre celui qui maximise $\deg^-_U(v, W, \omega)$;
        si $A(v) \cap S(v) \neq \varnothing$ **alors**
            Choisir la plus petite couleur $i \in A(v) \cap S(v)$ et insérer v dans V_i ;
        sinon
            Choisir la plus petite couleur $i \in A(v)$ et insérer v dans V_i ;
        Mettre à jour les ensembles $A(u)$ pour tout $u \in U$ tels que $i \in A(u)$, et enlever de U les sommets pour lesquels $A(u) = \varnothing$;
retourner La meilleure solution obtenue ;

La Figure 2.13 illustre la phase de prétraitement par coloration du pavage de Voronoï avec $\tau = 0$. Étant donné un ensemble de 10 stations de base, on commence par tracer le pavage de Voronoï associé. On en déduit le graphe dual de ce pavage, et celui-ci est coloré - 4 couleurs suffisent d'après le théorème des 4 couleurs. Plus exactement, la distance minimale entre deux sommets de même couleur est maximisée. Ensuite, on colore les régions de Voronoï de la même couleur que leur sommet, et l'on laisse tomber le graphe dual. La coloration du pavage obtenue permet alors de fixer les canaux prioritaires pour les usagers en bordure de ces cellules. Dans cet exemple, 4 couleurs sont utilisées, donc l'ensemble des canaux est divisé en 4 quarts de même cardinalité. Au cours de l'assignation des canaux, un usager se trouvant par exemple en bordure d'une cellule rouge prendra en priorité un canal appartenant au quart correspondant.

Tableau 4.6 Deuxième adaptation de l'algorithme RLF

Algorithme 12: RLF2

INITIALISATION :
$V_i = \varnothing$ pour $i = 1, \cdots, k$;
pour $\tau = 0, 0.1, 0.2, \cdots, 1$ **faire**
    Déterminer les ensembles de couleurs prioritaires $S(v)$ pour $v \in V$;
    COLORATION :
    pour $i = 1, \cdots, k$ **faire**
        Soit U l'ensemble des sommets non colorés et $U' \subseteq U$ le sous-ensemble des sommets non-colorés tels v tels que $i \in S(v)$; calculer $\deg_U^-(u, W, \omega)$ pour tout $u \in U$ et $\deg_{U'}^-(u, W, \omega)$ pour tout $u \in U'$; Poser $Z = \varnothing$ et $\deg_Z^-(u, W, \omega) = 0$ pour tout $u \in U$;
        si $U' \neq \varnothing$ **alors**
            Choisir un sommet $v \in U'$ qui maximise $\deg_{U'}^-(v, W, \omega)$;
        sinon
            Choisir un sommet $v \in U$ qui maximise $\deg_U^-(v, W, \omega)$;
        Insérer v dans V_i et déplacer dans Z tous les sommets de U (et donc de U') pour lesquels i n'est plus disponible ;
        tant que $U' \neq \varnothing$ **faire**
            Choisir un sommet $u \in U'$ qui maximise $\deg_Z^-(u, W, \omega)$; en cas d'égalité, choisir celui qui minimise $\deg_{U'}^-(u, W, \omega)$;
            Insérer u dans V_i, et déplacer dans Z tous les sommets de U pour lesquels i n'est plus disponible ;
        tant que $U \neq \varnothing$ **faire**
            Choisir un sommet $u \in U$ qui maximise $\deg_Z^-(u, W, \omega)$; en cas d'égalité, choisir celui qui minimise $\deg_U^-(u, W, \omega)$;
            Insérer u dans V_i, et déplacer dans Z tous les sommets de U pour lesquels i n'est plus disponible ;
retourner La meilleure solution obtenue ;

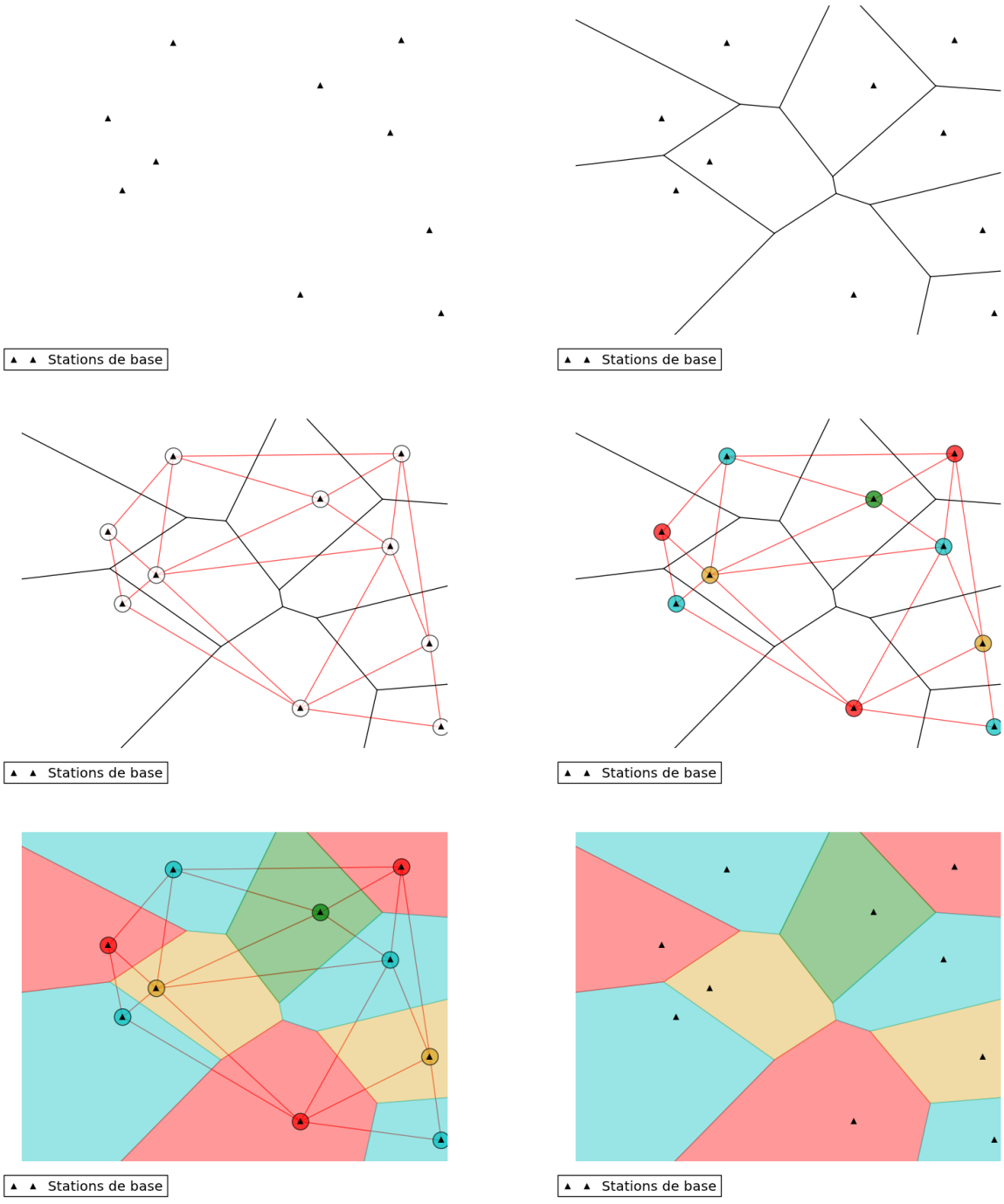


Figure 4.4 Détermination de canaux prioritaires à l'aide d'un pavage de Voronoï : 1. Répartition aléatoire de stations de base 2. Détermination du pavage de Voronoï correspondant 3. Création du graphe dual 4. Coloration du graphe dual 5. Coloration du pavage 6. Affectation de canaux prioritaires.

4.4.2 Couleurs super-disponibles

L'utilisation de couleurs prioritaires présentée précédemment n'est pas toujours suffisante pour éviter d'attribuer la même couleur à deux sommets u et v tels que $\omega(u,v)$ ou $\omega(v,u)$ soit sensiblement égal à $\theta\, W(v)$ ou $\theta\, W(u)$, respectivement. Voici donc une autre façon d'éviter ce problème, basée sur la définition suivante.

Définition 29. *Soit ρ un paramètre réel appartenant à l'intervalle $[0,1]$. Une couleur ℓ est dite super-disponible pour un sommet v, si en plus des contraintes (4.1), (4.2), les suivantes sont également satisfaites :*

$$\omega(u,v) \leq \rho(\theta W(v)) \quad \text{et} \quad \omega(v,u) \leq \rho(\theta W(u)) \qquad \forall u \in V_\ell.$$

Si $\rho = 0$, u et v ne peuvent pas prendre la couleur ℓ simultanément, et si $\rho = 1$, toutes les couleurs disponibles sont également super-disponibles. Les algorithmes appelés WP3, DSAT3, et RLF3 sont similaires aux algorithmes WP2, DSAT2, RLF2, avec pour unique différence que ce sont cette fois-ci les couleur super-disponibles qui sont assignées en priorité lorsque cela est possible. De même que pour WP2, DSAT2 et RLF2, les complexités algorithmiques ne sont pas affectées. Comme pour le paramètre $\tau \in [0,1]$ permettant de délimiter la zone centrale d'une cellule, les tests numériques de la section suivante montreront qu'il ne semble pas y avoir une valeur de ρ optimale, nous adoptons donc la même stratégie que précédemment : nous conserverons la meilleure solution parmi 11 essais avec les valeurs de ρ allant de 0 à 11.

4.5 Comment comparer ces algorithmes avec les méthodes d'affectations existantes ?

Avant de tester ces algorithmes numériquement, une question se pose : comment les comparer avec les méthodes d'affectation existantes utilisées dans l'industrie ? Malheureusement, les algorithmes utilisés ne sont pas publics, et il n'est donc pas possible de simuler une affectation réelle. Cependant, tel qu'évoqué dans le Chapitre 2, les fournisseurs de services mobiles se basent sur des modèles statiques et utilisent une technique appelée réutilisation de fréquence fractionnée [105]. Cette technique est communément utilisée comme référence dans la littérature [106, 107, 108] pour effectuer des comparaisons numériques.

Réutilisation de fréquence fractionnée

Tel que son nom l'indique, cette technique est basée sur l'utilisation de mêmes fréquences sur différentes régions géographiques. C'est en fait ce principe qui a servi de base pour développer les algorithmes améliorés WP2, DSAT2 et RLF2. Ainsi, on différencie les usagers

au centre d'une cellule de ceux en bordure, qui sont plus vulnérables aux interférences. À nouveau, les usagers en zone centrale ont accès à l'ensemble du spectre $F = \{1, \cdots, k\}$, tandis que les usagers en bordure d'une cellule C_p n'ont accès qu'à un sous ensemble de celui-ci, noté F_p. Mais en tant que modèle statique, les canaux sont préaffectés aux cellules de façon permanente, par conséquent le sous-ensemble F_p ne définit non pas un sous-ensemble de canaux prioritaires, mais le seul sous-ensemble possible pour un usager en bordure de C_p. De plus, une couleur ne peut être utilisée plus d'une fois par cellule. Autrement, dit si l'on note également C_p l'ensemble des usagers de la cellule de la station s_p et V_ℓ l'ensemble des usagers utilisant la couleur $\ell \in F$, alors il faut

$$| C_p \cap V_\ell | \leq 1 \qquad \forall p \in \{1, \cdots, t, \}, \ \forall \ell \in F. \tag{4.10}$$

Nous avons vu au Chapitre 2 que les modèles statiques sont basés sur des patrons génériques de réseaux réduits constitués de cellules hexagonales. Or, trois couleurs suffisent pour colorer un maillage hexagonal du plan de sorte que deux cellules ayant une frontière commune n'aient pas la même couleur. Par conséquent, il est toujours possible dans cette configuration de séparer l'ensemble des canaux disponibles F en 3 ensembles de même cardinalité, de façon à ce que les usagers en bordure de C_p aient accès au sous-ensemble $| F_p | = \frac{|F|}{3}$. Un exemple de patron est illustre à la Figure 4.5.

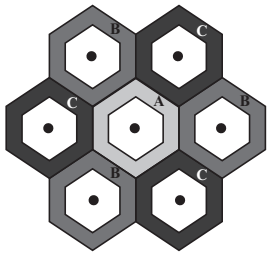


Figure 4.5 $F = A \cup B \cup C$; F_p est égal à A, B ou C ; $A \cap B = A \cap C = B \cap C = \varnothing$, $| A | = | B | = | C | = \frac{|F|}{3}$.

Il est clair qu'avec un tel modèle statique, au plus $|F| = k$ usagers par cellule peuvent être connectés simultanément au réseau. Voyons comment affiner cette borne.

Proposition 18. *Soit un réseau avec t stations de base et $|F| = k$ canaux. Soit x_p et y_p le nombre d'usagers au centre et en bordure de C_p, respectivement. Alors, le nombre maximum d'usagers pouvant être simultanément connectés au réseau par réutilisation de fréquence*

fractionnée est borné par

$$\sum_{p=1}^{t} \left(\min\{x_p, k\} + \min\left\{ \frac{k}{3}, y_p, \max\{0, k - x_p\} \right\} \right). \tag{4.11}$$

Démonstration. Cette borne peut être prouvée en calculant le flot maximum dans un graphe. Plus précisément, représentons les zones centrale et frontière d'une cellule C_p par deux sommets dans lesquels entrent respectivement x_p et y_p unités de flot, tel qu'illustré à la Figure 4.6 ci-dessous. Ajoutons également un sommet vers lequel convergent les flots provenant des deux zones. Enfin, imposons sur les arcs les limites supérieures de flot pouvant circuler, à savoir k pour l'arc sortant du sommet de la zone centrale, et $\frac{k}{3}$ pour celui sortant du sommet de la zone en bordure. Le nombre maximum d'usagers connectables au réseau via la station C_p correspond alors à un flot maximum dans ce graphe. La relation (4.11) s'éclaircit alors : le premier terme de la somme, $\min\{x_p, k\}$, correspond au flot de la branche supérieure, et le second, $\min\left\{\frac{k}{3}, y_p, \max\{0, k - x_p\}\right\}$, correspond à celui de la branche inférieure. On obtient la borne en sommant ces termes sur l'ensemble des cellules. □

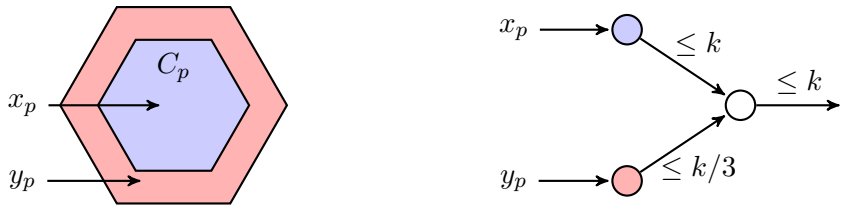


Figure 4.6 Détermination du nombre maximum d'usagers connectés dans une cellule par un flot maximum

Cette borne supérieure servira de référence pour les tests numériques dans la section suivante. Notons qu'il s'agit d'une borne très optimiste, car de nombreuses configurations réelles telles que celle illustrée à la Figure 4.7 ci-après ne permettent de fixer $\mid F_p \mid = \frac{k}{3}$ pour toute cellule C_p. De plus, il n'est pas garanti que la puissance d'un usager au centre n'interfère pas avec celles d'usagers en bordure de cellules voisines, autrement dit cette borne peut correspondre à une solution qui n'est pas nécessairement réalisable.

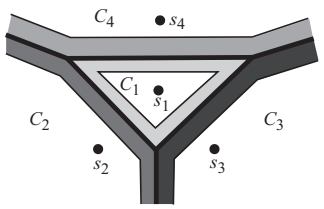


Figure 4.7 Une configuration de stations où F_p a au plus $\frac{|F|}{4}$ fréquences

4.6 Résultats numériques

Cette section est divisée en trois parties. Dans un premier temps, nous décrivons la façon dont nous avons généré des instances aléatoires pouvant représenter des réseaux réels. Deuxièmement, nous comparons les 9 algorithmes WPi, DSATi, RLFi ($i = 1, 2, 3$) avec la solution optimale sur de petites instances, et essayons de trouver des valeurs optimales des paramètres τ et ρ introduits à la section 4.4. Cette solution optimale est calculée par programmation linéaire en nombres entiers avec des méthodes telles que celles décrites au Chapitre 3. Enfin, nous comparons nos heuristiques à la borne supérieure décrite à la section précédente, donnant le nombre maximum d'usagers pouvant être connectés au réseau par réutilisation de fréquence fractionnée. Tous les tests sont effectués sur une machine 3.4 GHz Intel(R) Core(TM) i7-2600 CPU avec 16 GB de RAM.

4.6.1 Génération d'instances aléatoires

Les instances sont générées suivant les standards WINNER [109], norme de référence pour la propagation sans fil dans les systèmes de quatrième génération. Les positions des usagers et des stations de basent sont répartis aléatoirement dans le plan suivant une loi uniforme, en interdisant les configurations où deux stations sont trop proches l'une de l'autre. Les puissances (et donc les pondérations du graphe) sont calculées à partir de la formule décrite à la section 2.1, le facteur d'atténuation γ étant fixé à 4. La technologie actuelle exige un SINR supérieur ou égal à 6 dB (i.e. $\theta = 0.125$), et même $\theta = 0.25$ est un peu conservatif. Plus θ est grand, plus le modèle permettra d'avoir une meilleure efficacité spectrale ; nous l'avons fixé à $\theta = 0.25$. À l'échelle métropolitaine, les fournisseurs de services mobiles utilisent typiquement quelques dizaines de stations de base. Nous avons généré des instances avec $t = 10, 15$ et 25 stations. Enfin, d'après [110], les standards actuels tels que la LTE utilisent de l'ordre de 120 (168 exactement) ressources, ou canaux . Nous avons utilisé des valeurs de $k = 12, 36, 48, 120$.

4.6.2 Comparaison avec la solution optimale

Soit OPT la solution optimale d'une instance, obtenue par programmation linéaire en nombres entiers. Puisque OPT ne peut être obtenue que pour des instances de petite taille, ces tests sont effectués avec $t = 10$ stations de base, $n = 25, 30, 35, 40$ usagers, $k = 12$ canaux et $(\theta, \gamma) = (0.25, 4)$. Pour chaque combinaison des paramètres t, n, k, θ, γ, nous avons généré 25 instances aléatoires, soit un total de 100.

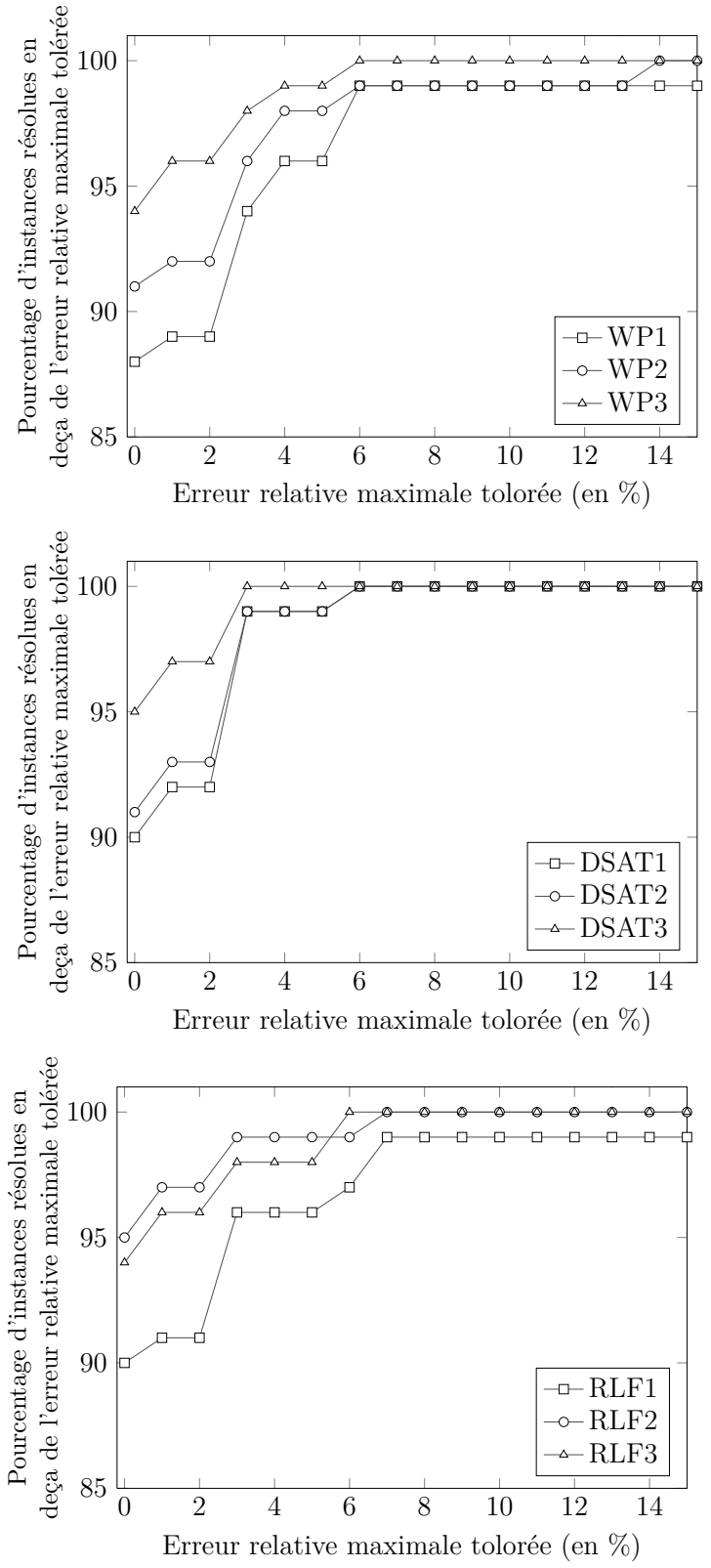


Figure 4.8 Comparaison des heuristiques avec la solution optimale

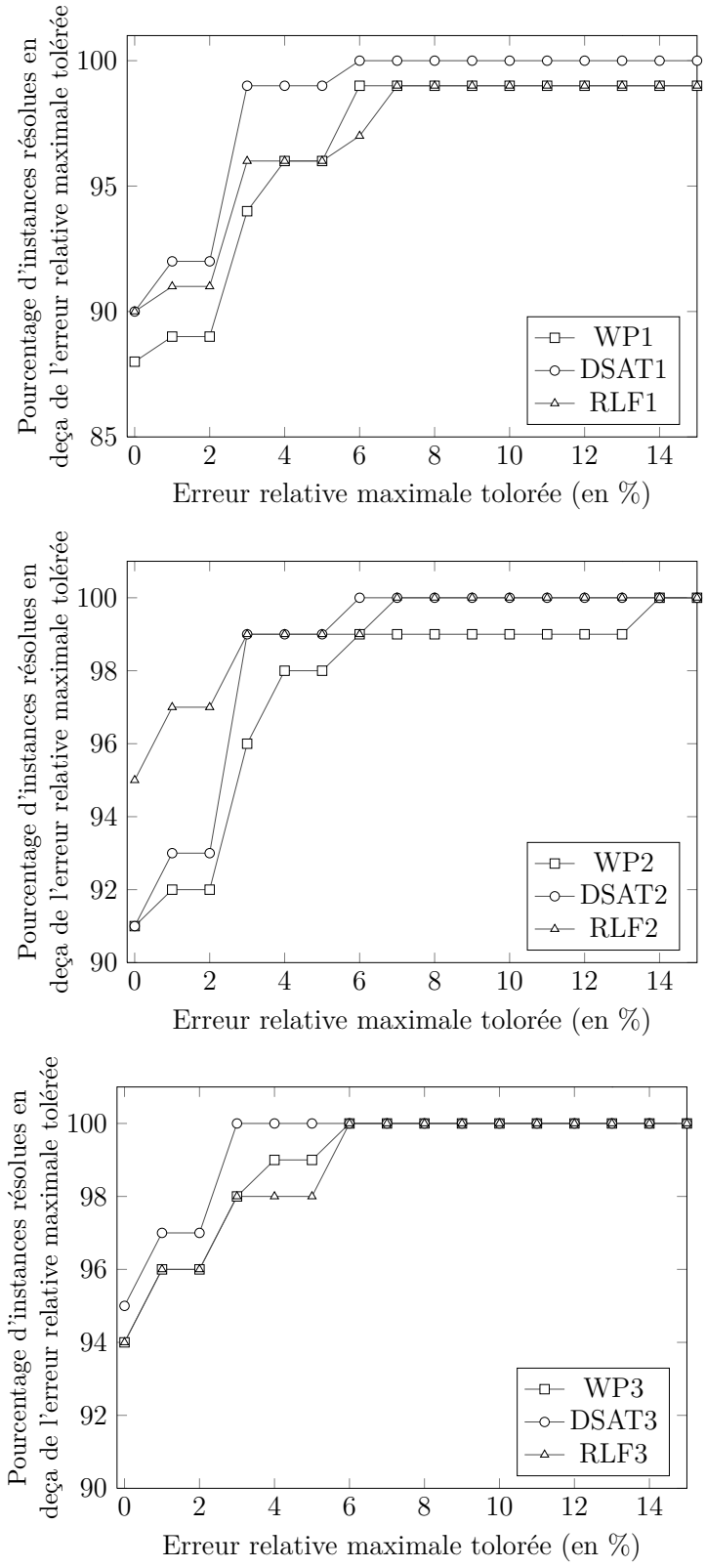


Figure 4.9 Comparaison des heuristiques avec la solution optimale

Les résultats moyens sont résumés dans la Figure 4.8. En abscisses, on considère différentes erreurs relatives possibles entre OPT et la solution fournie par les heuristiques, et en ordonnées on a le pourcentage d'instances résolues avec une erreur relative en deçà de l'abscisse correspondant. Par exemple, pour WP1, on observe que 88% des intances sont résolues de façon optimale, que l'erreur relative maximale ne dépasse pas 6% pour 99% des instances, et que 96% des instances sont résolues avec une erreur relative en deçà de 4%.

Il est clair que WP2 donne de meilleurs résultats que WP1, et que WP3 est encore meilleur que WP2, avec 94% de solutions optimales. L'erreur relative maximale de WP3 est de 6%. L'analyse est similaire pour les autres heuristiques : DSAT3 semble être la stratégie gagnante par rapport à DSAT1 et DSAT2, avec 95% de solutions optimales et une erreur relative qui n'excède pas 3%. Enfin, RLF3 produit 94% de solutions optimales, que l'on peut comparer avec les 90% de RLF1 et 95% de RLF2. L'erreur relative de RLF3 par rapport à OPT ne dépasse jamais 6%.

Dans les trois graphiques de la Figure 4.9, les heuristiques WPi, DSATi and RLFi, ($i = 1, 2, 3$) sont comparées entre elles. Il s'avère que DSAT1 donne de meilleurs résultats que WP1 et RLF1, alors que RLF2 réalise de meilleures performances que WP2 et DSAT2, et DSAT3 bat WP3 et RLF3. Remarquons également que l'erreur relative maximale obtenue par DSATi est toujours inférieure à celles de WPi et RLFi : 6% pour DSAT1 et DSAT2, et 3% pour DSAT3. Ces graphiques mettent bien en évidence le fait que ces heuristiques peuvent atteindre l'optimalité pour un grand nombre d'instances. Toutes, en dehors de WP1, ont au moins 90% de solutions optimales. De plus, les améliorations proposées à la Section 4.4 sont effectivement plus efficaces que les adaptions initiales.

Si DSAT3 peut apparaître comme la stratégie gagnante, il est important de mentionner que WP3 est aussi attrayant, car sa complexité est en $\mathcal{O}(n^2)$, alors que celles de DSAT3 et RLF3 est en $\mathcal{O}(n^3)$. Cela apparaît clairement dans le Tableau 4.7 où sont résumés les temps de calculs (en secondes), pour des réseaux de grande taille ayant entre $n = 500$ et $n = 1100$ usagers, $k = 120$ canaux, $t = 25$ stations de base, et (θ, γ) =(0.25,4). Remarquons aussi que si les versions 2 et 3 de nos heuristiques sont exécutées 11 fois (pour 11 valeurs différentes des paramètres τ et ρ), les temps sont comparables à ceux de la version 1. Ceci s'explique par le fait que le choix de la couleur à assigner à un sommet est plus limité et donc s'effectue plus rapidement.

Tableau 4.7 Temps CPU moyen (en secondes) avec $(\theta, \gamma) = (0.25, 4)$, $t = 25$ stations, $k = 120$ couleurs

n	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
500	0.20	0.22	0.30	0.78	0.83	0.85	7.10	7.10	8.22
550	0.30	0.30	0.40	0.98	1.00	1.03	9.40	9.40	11.52
600	0.33	0.42	0.48	1.28	1.32	1.37	13.65	13.65	16.28
650	0.37	0.43	0.58	1.52	1.50	1.62	16.28	16.28	19.68
700	0.42	0.43	0.60	1.53	1.65	1.62	20.05	20.05	22.30
750	0.50	0.57	0.68	1.88	1.80	1.87	23.15	23.15	28.45
800	0.58	0.67	0.83	2.27	2.32	2.48	32.23	32.23	34.93
850	0.70	0.75	0.95	2.68	2.60	2.72	38.33	38.33	36.63
900	0.73	0.73	0.92	2.67	2.62	2.90	40.87	40.87	38.63
950	0.77	0.82	1.02	2.63	2.50	2.78	55.35	55.35	48.33
1000	0.90	0.87	1.03	3.05	3.13	3.37	59.75	59.75	45.78
1050	0.95	1.00	1.20	3.67	3.58	3.97	66.20	66.20	45.22
1100	1.10	1.12	1.32	3.97	4.03	4.58	79.75	79.75	53.63

Ajustement des paramètres τ et ρ

Les versions 2 et 3 des heuristiques proposées dépendent des paramètres $\tau \in [0, 1]$ et $\rho \in [0, 1]$, respectivement. Nous avons effectué des tests afin d'essayer de déterminer si une valeur spécifique permettait d'avoir les meilleurs résultats. Les tests sont à nouveau réalisés avec $t = 10$, $n = 25, 30, 35, 40$, $k = 12$, $\theta = 0.25$ et $\gamma = 4$, et 25 instances sont générées pour chaque combinaison t, n, k, θ, γ, pour un total de 100 instances. Les résultats moyens donnés par WP2 et WP3 sont représentés aux Figures 4.10 et 4.11, pour τ et ρ égaux à $0, 0.25, 0.5, 0.75$ et 1. Donc, plutôt que de retourner la meilleure solution parmi les 11 exécutions de l'algorithme (comme à la Figure 4.8), nous analysons les résultats obtenus pour chacune des valeurs de τ et ρ.

On observe qu'environ 80% de solutions optimales sont obtenues avec $\tau \geq 0.75$, alors que 73% sont obtenues avec $\tau = 0.5$, et environ 70% avec $\tau \leq 0.25$. Cependant, des tests effectués sur de plus grandes instances semblent indiquer qu'il n'existe pas de valeur spécifique sortant du lot. Par exemple, ces tests ont montré que $\tau = 0.75$ produit de meilleurs résultats que $\tau = 0.5$ 27% du temps, mais de moins bons 13% du temps. Pour le paramètre ρ, les résultats représentés à la Figure 4.11 sont similaires : il ne semble pas y avoir une valeur spécifique qui donne systématiquement de meilleurs résultats.

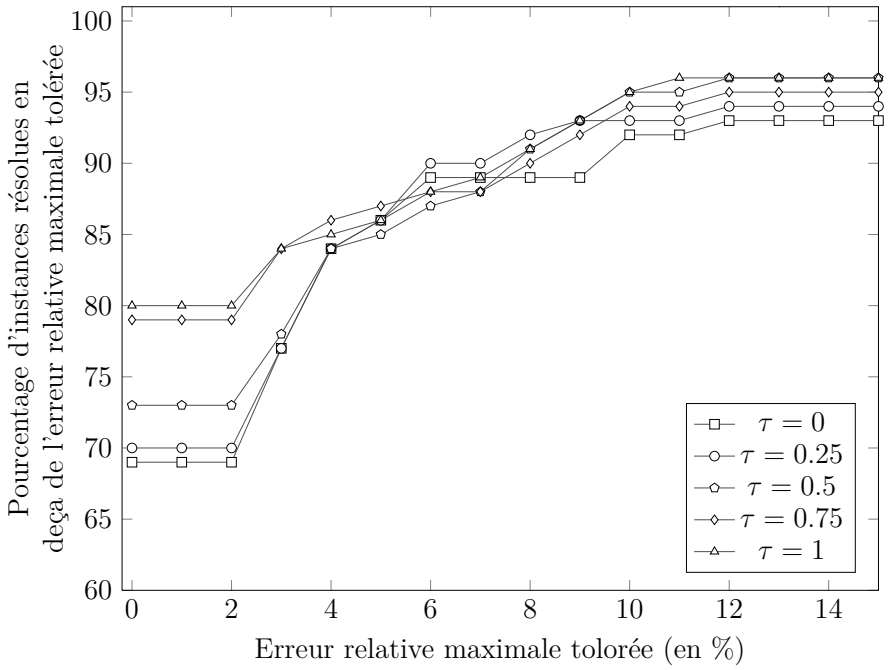


Figure 4.10 Performances de WP2 pour différentes valeurs de τ.

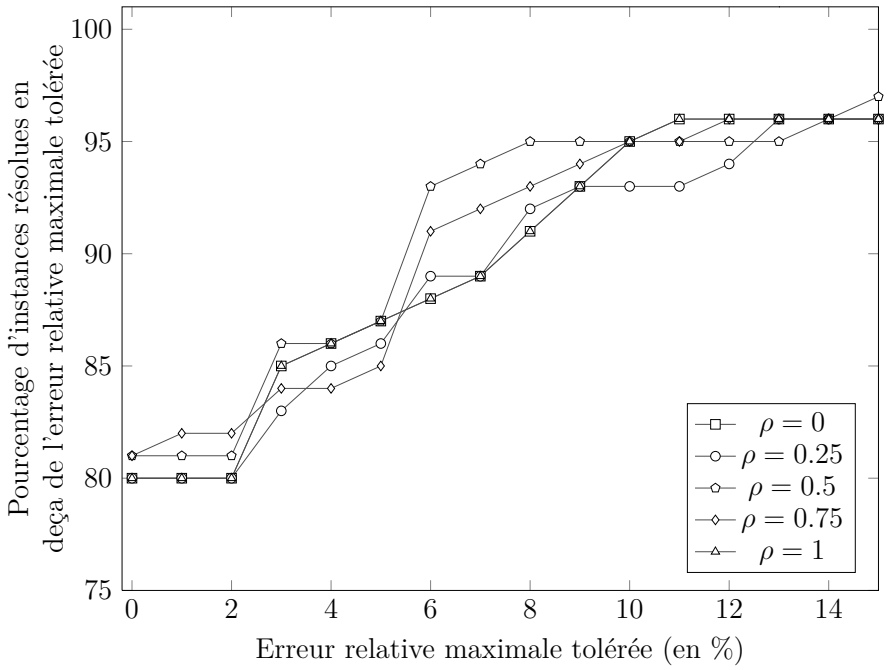


Figure 4.11 Performances de WP3 pour différentes valeurs de ρ.

4.6.3 Comparaison avec la réutilisation de fréquence fractionnée des modèles statiques

On compare ici les performances des heuristiques avec la réutilisation de fréquence fractionnée. Nous avons vu précédemment que l'on pouvait calculer une borne supérieure sur la couverture (en %) obtenue par cette technique :

$$\frac{100}{n}\sum_{p=1}^{t}\left(\min\{x_p,k\}+\min\left\{\frac{k}{3},y_p,\max\{0,k-x_p\}\right\}\right).$$

Cette borne supérieure, notée FFR3, est calculée en fixant τ à $\frac{\theta}{2}$ (rappelons que θ est fixé à 0.25) afin de déterminer x_p et y_p dans chaque cellule C_p. Si θ était nul (si aucune interférence n'était tolérée), alors tous les usagers seraient considérés comme étant en bordure de cellule et n'auraient accès qu'au tiers du spectre disponible selon leur emplacement. Inversement, si θ prenait la valeur 1, le seuil de tolérance serait très élevé, et chaque usager pourrait prendre n'importe quel canal non utilisé dans sa cellule (ce scénario n'est pas réaliste).

Les résultats pour différents couples (t,k) sont présentés dans les Tableaux 4.8 - 4.12. Chaque entrée du tableau correspond à la couverture moyenne (i.e. le pourcentage d'usagers qui ont pu avoir un canal) calculée sur 30 instances aléatoires. Des bornes sur la valeur optimale OPT sont également données lorsque $t \leq 15$ et $k \leq 48$. Plus précisément, $\underline{z}$ et $\overline{z}$ représentent la meilleure solution réalisable et la plus petite borne supérieure obtenue par le solveur CPLEX à l'issue de 30 secondes de calcul. Si $\underline{z} = \overline{z}$, cela signifie que la solution optimale est atteinte. La Figure 4.12 montre la distribution statistique de ces résultats pour les triplets (n,t,k)=(30,10,12), (80,10,24), (175,15,36) et (225,15,48), avec les conventions classiques de boîtes à moustaches. En prime figurent les résultats de FFR4, la borne FFR3 où $\frac{k}{3}$ est remplacé par $\frac{k}{4}$; les écarts sont encore plus flagrants.

Pour les réseaux à $t = 10$ stations de base et $k = 12$ canaux, on observe dans le Tableau 4.8 que non seulement WPi, DSATi et RLFi $(i = 1, 2, 3)$ donnent une meilleure couverture que la borne optimiste FFR3, mais les solutions sont proches de l'optimalité. Par exemple, pour $n = 30$ usagers, FFR3 donne une couverture moyenne de 96.3%, tandis que les heuristiques produisent une solution avec au moins 99.5% d'usagers connectés au réseau, la solution optimale étant à 99.7%. Pour $n = 200$ usagers et $(t,k) = (15, 48)$ (voir le Tableau 4.11), toutes les heuristiques améliorent la couverture de FFR3 (97.9%) d'environ 2%, soit 4 usagers de plus connectés au réseau. Les conclusions pour les plus grandes instances sont analogues. Par exemple, pour $n = 800$ usagers et $(t,k) = (25, 120)$ (voir le Tableau 4.12), FFR3 donne une couverture de 97.9%, tandis que les heuristiques produisent des solutions où la couverture varie de 98.1% (WP1) à 99.9% (DSAT3), soit jusqu'à 16 usagers supplémentaires connectés au

réseau. Les distributions statistiques (Figure 4.12) montrent que ces résultats sont robustes.

Une comparaison plus visuelle est donnée à la Figure 4.13. Pour gagner en lisibilité, nous n'avons représenté que les troisièmes versions WP3, DSAT3 et RLF3 des algorithmes. La meilleure borne supérieure $\bar{z}$ est également omise lorsqu'elle est sensiblement égale à 100%. Les standards de téléphonie mobile doivent être en mesure de garantir une couverture au moins égale à 98% ; ce seuil noté CBP [2] est représenté par une ligne en pointillée. Pour le couple $(t, k) = (10, 12)$, on observe que ce seuil est dépassé par FFR3 avec 25 usagers, alors que 10 usagers supplémentaires peuvent se connecter au réseau avec WP3, DSAT3 et RLF3. Pour $(t, k) = (15, 48)$, le seuil est dépassé avec 200 usagers pour FFR3, et le gain est de 20 usagers supplémentaires pour les trois heuristiques. Les résultats obtenus sur des réseaux de grande taille avec $t = 25$ stations et $k = 120$ canaux donnent des résultats intéressants (voir le Tableau 4.12 et la Figure 4.13). On constate tout d'abord que le seuil des 98% est dépassé par FFR3 avec plus de 850 usagers, alors que plus de 100 usagers supplémentaires sont connectés au réseau avec WP3, DSAT3, ou RLF3.

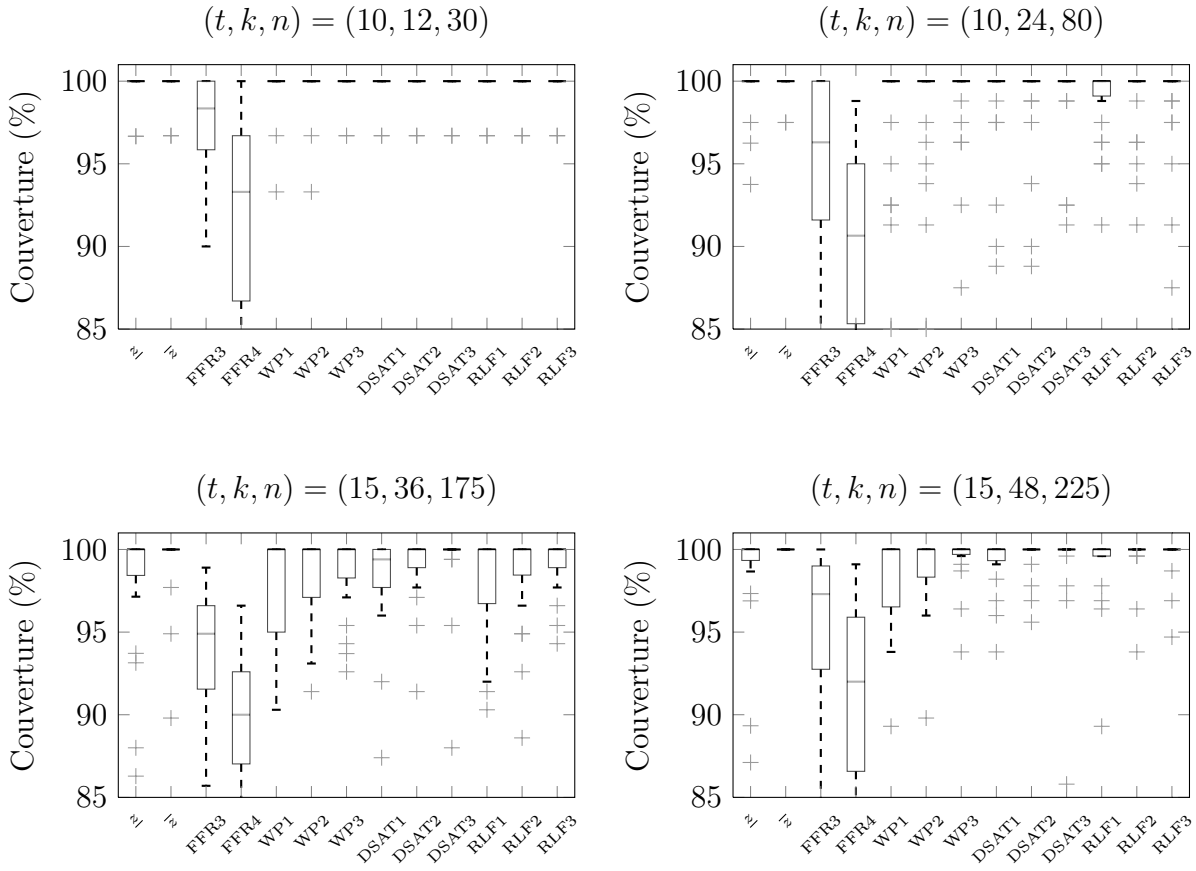


Figure 4.12 Boîtes à moustaches de la couverture avec $(\theta, \gamma) = (0.25, 4)$

2. Call Blocking Probability of 2%, en anglais.

Tableau 4.8 Couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$, $t = 10$ stations, $k = 12$ canaux

n	$\underline{z}$	$\overline{z}$	FFR3	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
15	100.0	100.0	99.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
20	100.0	100.0	99.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
25	100.0	100.0	97.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
30	99.7	99.7	96.3	99.5	99.5	99.7	99.7	99.7	99.7	99.7	99.7	99.7
35	99.1	99.1	93.3	97.6	98.3	98.1	98.6	98.7	98.7	98.1	98.4	98.6
40	98.5	98.9	90.5	95.9	96.5	96.9	96.5	97.1	96.9	96.0	97.8	97.3
45	96.7	97.0	88.8	88.2	89.7	90.9	91.3	92.2	92.0	89.0	92.2	91.9
50	96.5	96.8	89.2	90.2	91.6	92.8	93.2	94.1	93.8	90.8	93.1	92.8

Tableau 4.9 Couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$, $t = 10$ stations, $k = 24$ canaux

n	$\underline{z}$	$\overline{z}$	FFR3	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
60	99.7	99.8	95.1	99.3	99.4	99.5	99.4	99.6	99.7	99.2	99.4	99.3
65	99.8	99.8	95.6	99.5	99.5	99.5	99.7	99.7	99.7	99.4	99.5	99.5
70	100.0	100.0	96.2	99.8	100.0	100.0	99.9	100.0	100.0	99.9	100.0	100.0
75	99.5	99.7	96.3	98.1	98.6	99.0	98.9	99.3	99.3	98.5	98.9	98.9
80	99.6	99.8	94.9	97.8	98.1	98.4	98.8	98.9	99.1	98.3	99.0	98.9
85	98.9	99.5	93.6	97.0	97.3	97.8	97.3	97.8	97.6	97.4	97.9	98.1
90	98.4	99.0	91.3	94.0	94.6	95.4	95.3	96.1	96.4	94.3	96.2	96.3
95	97.3	98.5	90.1	92.7	94.1	94.9	94.9	96.0	96.1	93.2	95.2	95.2
100	97.2	98.6	89.7	91.3	92.9	94.7	94.3	95.7	96.0	91.8	94.5	95.1
105	92.9	96.3	89.4	84.2	85.9	88.5	88.8	90.2	89.8	85.2	88.6	88.7

Tableau 4.10 Couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$, $t = 15$ stations, $k = 36$ canaux

n	$\underline{z}$	$\overline{z}$	FFR3	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
75	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
100	100.0	100.0	99.4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
125	99.9	100.0	98.7	99.9	99.9	99.9	100.0	100.0	100.0	99.9	99.9	100.0
150	99.2	99.8	94.8	98.5	98.9	99.2	99.1	99.6	99.5	99.0	99.2	99.3
175	98.3	99.4	94.2	96.0	96.5	97.2	97.0	97.8	98.0	96.3	98.0	97.6
200	95.0	99.9	93.2	92.6	93.4	94.9	92.7	94.5	95.3	93.3	95.3	95.7
225	88.7	100.0	89.6	80.4	82.6	88.1	84.9	87.2	88.6	81.1	85.7	89.4

Remarquons que les versions 1 et 2 des algorithmes ne sont pas aussi efficaces que la version 3 lorsque la solution optimale s'écarte de 100%. Par exemple, la couverture moyenne de WP3 pour 1100 usagers est de 91.6%, alors que celles de WP1 et WP2 sont environ égales à 78.8% et 80.2%, respectivement, soit une perte de plus de 10% ou encore un manque à gagner de plus de 100 usagers. Ceci peut s'expliquer de la façon suivante : lorsque la solution optimale s'écarte des 100%, il semble raisonnable de rejeter les sommets de degré entrant élevé (i.e. ceux qui recoivent les plus fortes interférences). Or, les heuristiques (versions 1 et 2 en particulier) sont justement conçues pour traiter ces sommets en priorité. Les algorithmes WP3, DSAT3, et RLF3 sont mieux équipés pour faire face à cela : des usagers qui interfèrent fortement entre eux ont peu de chances d'avoir le même canal, laissant ainsi plus de place aux autres.

Tableau 4.11 Couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$, $t = 15$ stations, $k = 48$ canaux

n	$\underline{z}$	$\overline{z}$	FFR3	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
75	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
100	100.0	100.0	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
125	100.0	100.0	99.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
150	100.0	100.0	98.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
175	100.0	100.0	98.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
200	99.7	100.0	97.9	99.6	99.7	99.8	99.8	99.9	99.9	99.8	99.8	99.8
225	95.0	100.0	95.2	95.3	96.0	97.1	96.7	97.0	97.1	96.2	97.5	97.2
250	0.0	100.0	93.2	92.1	92.8	94.1	93.7	94.3	94.8	92.7	94.7	94.4

Tableau 4.12 Couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$, $t = 25$ stations, $k = 120$ canaux

n	FFR3	WP1	WP2	WP3	DSAT1	DSAT2	DSAT3	RLF1	RLF2	RLF3
500	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
550	99.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
600	99.4	99.9	99.9	100.0	100.0	100.0	100.0	100.0	100.0	100.0
650	99.5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
700	98.9	99.5	99.7	99.8	99.8	99.9	99.9	99.4	99.7	99.9
750	99.0	99.7	99.8	99.9	99.7	99.9	100.0	99.8	99.9	100.0
800	97.9	98.1	98.7	99.5	98.2	98.9	99.9	98.2	99.0	99.7
850	98.1	98.4	98.6	99.5	98.2	98.6	99.2	98.2	98.7	99.6
900	97.0	95.8	96.5	98.9	95.1	96.4	98.3	95.9	96.9	99.2
950	96.9	93.8	94.8	98.5	92.5	94.7	98.9	93.1	95.8	98.8
1000	95.5	90.2	91.3	97.4	90.3	92.5	97.1	88.2	91.5	98.0
1050	94.2	83.5	84.7	93.9	83.6	85.9	90.8	82.0	87.6	94.6
1100	92.8	78.8	80.2	91.6	79.5	82.2	87.9	76.8	81.0	92.9

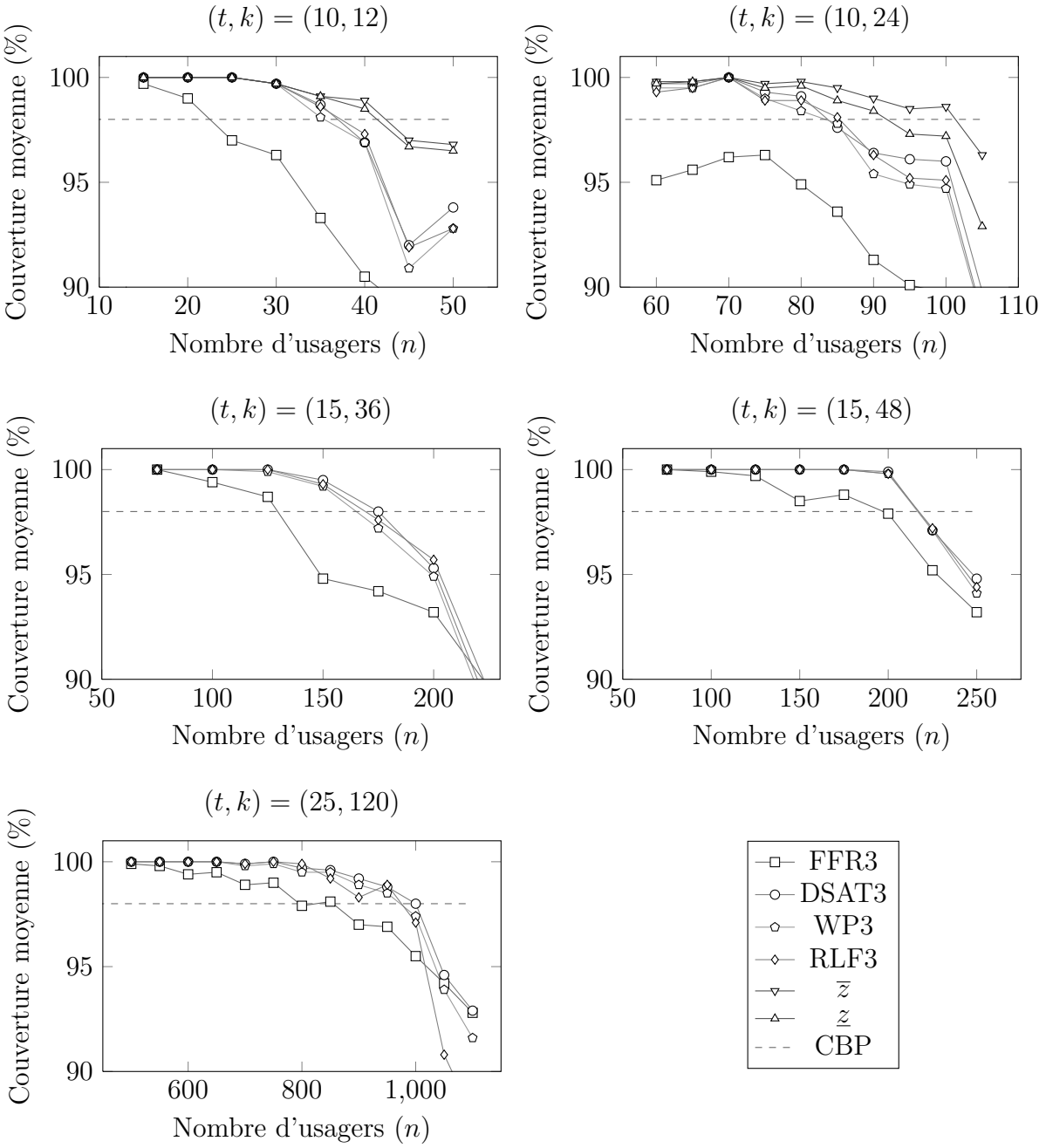


Figure 4.13 Évolution de la couverture moyenne avec $(\theta, \gamma) = (0.25, 4)$

Nous avons également effectués des tests en replaçant $\mid F_p \mid = \frac{k}{3}$ par $\mid F_p \mid = \frac{k}{4}$ dans la borne FFR3, de façon à avoir une borne supérieure moins optimiste et plus réaliste sur ce que peut donner la réutilisation de fréquence fractionnée. Les résultats sont alors d'autant plus flagrants : le seuil des 98% est dépassé par FFR3 avec 650 usagers ; RLF3 améliore ce résultat par plus de 50% avec plus 1000 usagers connectés au réseau.

4.7 Conclusion

Ce chapitre était dédié à la résolution approchée du (k, W, ω, θ)-PGSI. Pour ce faire nous avons adapté trois des heuristiques de coloration les plus populaires pour la coloration classique visant à approcher le paramètre $\chi(G)$. Nous avons généralisé les notions de degré entrant et de degré de saturation pour les graphes complets, pondérés et orientés, qui sont les ingrédients principaux de ces algorithmes.

Les heuristiques développées ont été analysées théoriquement via les notions de complexité, de garantie, et de plus petit graphe sous-optimal, et numériquement, via des comparaisons avec la solution optimale obtenue par programmation en nombres entiers, et avec une borne supérieure sur ce que peut donner la réutilisation de fréquence fractionnée sur un modèle statique. Les tests numériques montrent que le modèle proposé, couplé à ces heuristiques, permet de desservir plus d'usagers que la réutilisation de fréquence fractionnée, et ceci avec des temps de calculs attrayants, en particulier pour WP3.

Les algorithmes ont été implantés en langage Python, qui est un langage dit de haut niveau, typiquement utilisé pour développer des prototypes. Bien sûr, il serait possible de diminuer les temps de calculs en utilisant des langages de plus bas niveau. L'étude de complexité donne néanmoins un ordre de grandeur sur les pires temps de calculs auxquels ont pourrait s'attendre, indépendamment du langage de programmation utilisé.

Le modèle résolu ici est statique, c'est-à-dire qu'il considère les usagers à un instant donné et permet de maximiser la connectivité du réseau à cet instant. Une question légitime se pose : comment étendre ce modèle sur un horizon de temps, lorsque les usagers se déplacent, apparaissent et disparaissent du réseau ? Est-il judicieux d'exécuter les heuristiques séquentiellement, à chaque fois que le réseau change de topologie ? Cela induirait nécessairement des changements de canaux au cours du temps : ceci est-il permis ? Peut-on imaginer un procédé qui « répare » une affectation donnée qui n'est plus réalisable due à un changement de topologie ? Nous répondons à ces questions dans le chapitre qui suit.

CHAPITRE 5 EXTENSION DU MODÈLE SUR UN HORIZON DE TEMPS : COLORATION ON-LINE

Dans ce chapitre, le modèle d'affectation de canaux est étendu sur un horizon de temps. Ceci signifie que les usagers apparaissent dans le réseau au fur et à mesure et modifient la structure du graphe à chaque période de temps. La difficulté principale réside dans le fait que les décisions sont prises sans connaissance totale de la topologie du réseau à venir, et sont en principe irrévocables. Cette problématique s'inscrit dans le cadre de la coloration dite on-line.

Dans un premier temps, nous allons analyser les meilleures garanties de performance possibles pour ce problème, spécifiquement nous généraliserons des résultats de [91] évoqués dans le Chapitre 2, le cas particulier avec $k = 1$ couleur. Nous proposerons alors une série d'algorithmes s'appuyant sur ceux du Chapitre 4 dans lesquels sont introduites de nouvelles règles de priorité. Enfin, nous considérerons une version relaxée du problème dans laquelle il est permis de revenir sur une décision prise dans le passé, moyennant une pénalité. Ceci donne lieu à un problème bi-objectif, que nous traiterons avec deux approches distinctes : par des algorithmes dits de réparation, et par recherche locale. Une série de tests numériques sont présentés et analysés pour évaluer la performance des algorithmes proposés.

De même que dans le chapitre précédent, nous allons temporairement laisser tomber les pondérations sur les graphes, ainsi que le caractère θ-impropre des colorations : nous considérons donc dans un premier temps le $(k, 1, 1, 0)$-PGSI. L'utilisation de ces cas particuliers permet simplement de gagner en lisibilité et donc en clarté. De plus, le fait de travailler avec des graphes classiques permet de réaliser les tests numériques sur les instances DIMACS dont les paramètres sont connus, ce qui facilite l'analyse des résultats obtenus. Nous rappellerons dans un deuxième temps pourquoi les résultats théoriques sont généralisables pour des valeurs de θ quelconques, et développerons des algorithmes spécifiquement dédiés au (k, W, ω, θ)-PGSI.

5.1 Analyse de compétitivité on-line

Les notations utilisées demeurent les mêmes, à savoir n représente la cardinalité de l'ensemble des sommets V et k le nombre de couleurs disponibles. On note $A(G)$ la valeur de la solution fournie par un algorithme A sur un graphe G. Rappelons que la qualité d'une solution obtenue par un algorithme on-line A peut être évaluée par le ratio $A(G)/\alpha_k(G)$, où $\alpha_k(G)$ est la solution optimale (a fortiori off-line). Le premier scénario considéré est le suivant : à chaque

itération, un sommet ainsi que ses arêtes incidentes sont révélés. Il y a donc autant de sommets dans le graphe que d'itérations.

Proposition 19. *Aucun algorithme on-line ne peut garantir un ratio de performance strictement plus grand que* $\frac{k}{n-1}$.

Démonstration. Tel qu'expliqué au Chapitre 2, il suffit pour prouver cela de mettre en évidence l'existence d'un algorithme (le *spoiler*) qui garantit que $A(G)/\alpha_k(G) \leq k/(n-1)$, quelque soient les choix de l'algorithme on-line A. Un tel algorithme est résumé dans le Tableau 5.1 ci-dessous, et un exemple de son application est illustré à la Figure 5.1 lorsque $k = 2$ et $n = 5$.

Tableau 5.1 Algorithme spoiler de la Proposition 19

Algorithme 13: Spoiler

```
1 INITIALISATION :
2 Révéler un sommet isolé v_1 ;
3 Poser t = 1 ;
4 Le sommet v_1 est coloré [1] ou rejeté par A ;
5 RÉVÉLATION :
6 tant que t ≤ n faire
7     Révéler v_{t+1} de sorte qu'il soit uniquement relié aux sommets déjà colorés ;
8     Le sommet v_{t+1} est coloré ou rejeté par A ;
9     t ← t + 1 ;
```

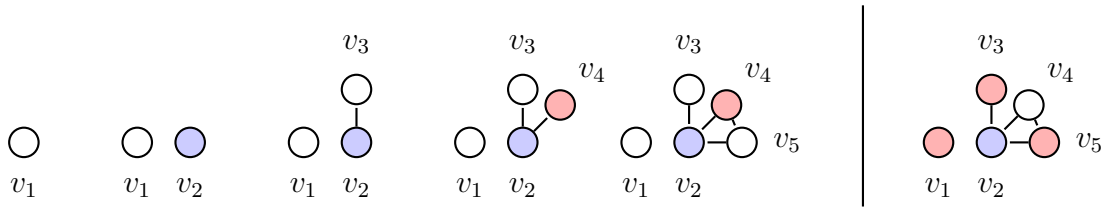


Figure 5.1 $A(G) = 2$ (à gauche), $\alpha_2(G) = 4$ (à droite)

Si aucun sommet n'est jamais coloré, le résultat est trivial : $A(G)$=0. Sinon, dès qu'un sommet est coloré, tous les sommets suivants lui sont adjacents, par conséquent une couleur ne peut servir plus d'une fois. Il s'ensuit que A ne peut générer un sous-graphe induit de cardinalité strictement plus grande que k, alors que $\alpha_k(G) = n - 1$. En effet, il suffit de prendre la

1. L'algorithme A est libre de choisir n'importe quelle couleur disponible. Ceci reste valable pour la suite.

coloration donnée par A, en excluant le dernier sommet coloré. La couleur restante est alors utilisée pour colorer tous les autres sommets non colorés. Dans l'exemple de la Figure 5.1, la solution optimale est obtenue utilisant la couleur rouge du sommet v_4 pour colorer les sommets v_1, v_3 et v_5, et en laissant le sommet v_2 coloré en bleu. □

Sous ces hypothèses, aucun algorithme on-line ne peut donc garantir un ratio de performance meilleur que $\mathcal{O}(\frac{k}{n})$. Il est clair que l'algorithme qui colore les k premiers sommets avec une couleur différente atteint asymptotiquement cette borne.

Supposons maintenant qu'à chaque itération, un ensemble de sommets apparaissent (ainsi que leurs arêtes incidentes), et qu'à l'issue de $|T| < n$ itérations, le graphe est entièrement révélé.

Proposition 20. *Si $k \leq \sqrt{n} - 1$, alors aucun algorithme on-line ne peut garantir un ratio de performance strictement plus grand que $\dfrac{k}{\sqrt{n} - 1}$.*

Démonstration. L'algorithme résumé dans le Tableau 5.2 génère un graphe d'ordre n tel que $A(G)/\alpha_k(G) \leq k/(\sqrt{n} - 1)$, quelque soient les choix de l'algorithme A. Un exemple de son application est illustré à la Figure 5.2 pour $n = 10$ et $k = 2$, pour deux configurations possibles ; les autres configurations sont semblables.

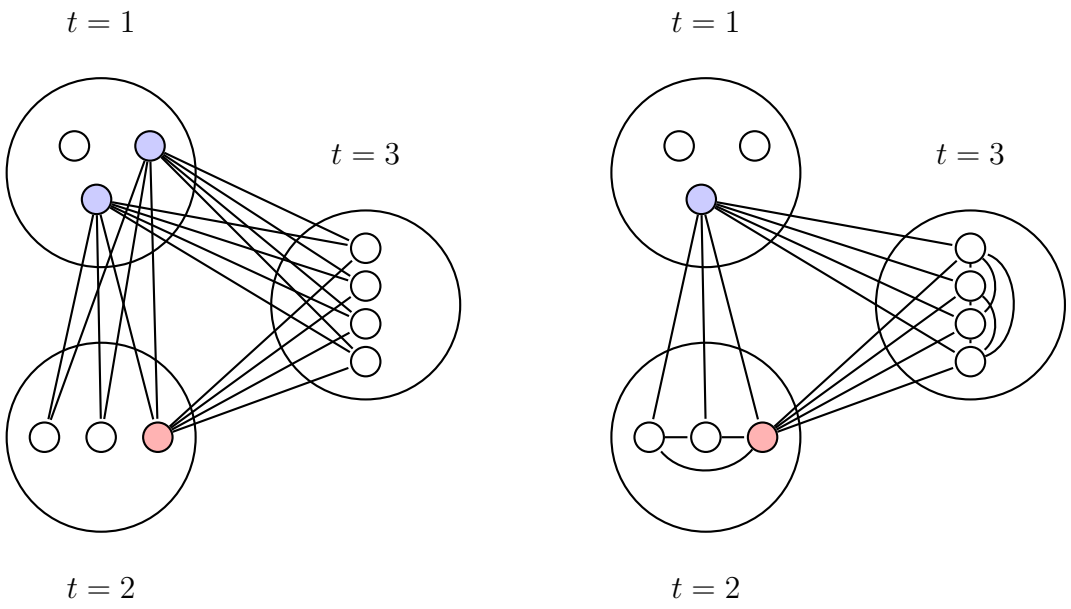


Figure 5.2 Exemples avec $k = 2$, $n = 10$; à gauche $\frac{A(G)}{\alpha_k(G)} \leq \frac{k\lfloor\sqrt{n}\rfloor}{n-\lfloor\sqrt{n}\rfloor} \leq \frac{k}{\sqrt{n}-1}$, à droite $\frac{A(G)}{\alpha_k(G)} \leq \frac{k}{\lfloor\sqrt{n}\rfloor} \leq \frac{k}{\sqrt{n}-1}$

Tableau 5.2 Algorithme spoiler de la Proposition 20

Algorithme 14: Spoiler

```
1  INITIALISATION :
2  Révéler un ensemble de ⌊√n⌋ sommets isolés ;
3  Poser t = 1 ;
4  Chacun des sommets est coloré ou rejeté par A ;
5  RÉVÉLATION :
6  si au moins 2 sommets sont colorés alors
7      tant que il reste au moins une couleur disponible et t ≤ ⌊n/⌊√n⌋⌋ faire
8          Révéler un ensemble de ⌊√n⌋ sommets non reliés entre eux, et reliés aux
           sommets déjà colorés ;
9          Chacun des sommets est coloré ou rejeté par A ;
10         t ← t + 1 ;
11     Révéler le restant du graphe, de sorte que les nouveaux sommets ne soient pas
       reliés entre eux, et reliés aux sommets déjà colorés ;
12     Chacun des sommets est coloré ou rejeté par A ;
13 sinon
14     tant que il reste au moins une couleur disponible et t ≤ k faire
15         Révéler un ensemble de ⌊√n⌋ sommets reliés entre eux, et aux sommets déjà
           colorés ;
16         Chacun des sommets est coloré ou rejeté par A ;
17         t ← t + 1 ;
18     Révéler le restant du graphe, de sorte que les nouveaux sommets soient reliés entre
       eux, et aux sommets déjà colorés ;
19     Chacun des sommets est coloré ou rejeté par A ;
```

Cas 1 : à l'issue de la première itération, au moins deux sommets sont colorés (lignes 6 - 12). L'algorithme spoiler révèle alors des ensembles de $\lfloor\sqrt{n}\rfloor$ sommets non reliés entre eux, mais reliés aux sommets colorés. Ainsi, après k itérations, au plus $k\lfloor\sqrt{n}\rfloor$ sommets peuvent être colorés (lignes 7 - 10). Si les k couleurs sont épuisées, alors aucun autre sommet ne peut être coloré dans les itérations qui suivent, et il s'ensuit que $A(G) \leq k\lfloor\sqrt{n}\rfloor$. Sinon, c'est qu'après $\left\lfloor \frac{n}{\lfloor\sqrt{n}\rfloor} \right\rfloor$ itérations, il reste au moins une couleur de disponible, et au plus $(k-1)\lfloor\sqrt{n}\rfloor$ sommets ont été colorés. Puisque le restant du graphe contient moins de $\lfloor\sqrt{n}\rfloor$ sommets (lignes 11 - 12), on a $A(G) \leq k\lfloor\sqrt{n}\rfloor$. Dans tous les cas, au plus $k\lfloor\sqrt{n}\rfloor$ sommets ont été colorés. Or, il est toujours possible de colorer au moins $n - \lfloor\sqrt{n}\rfloor$ sommets (par exemple, en attribuant une couleur par paquet, excepté celui de la première itération). On a donc

$$\frac{A(G)}{\alpha_k(G)} \leq \frac{k\lfloor\sqrt{n}\rfloor}{n - \lfloor\sqrt{n}\rfloor} = \frac{k}{\frac{n}{\lfloor\sqrt{n}\rfloor} - 1} \leq \frac{k}{\sqrt{n} - 1}.$$

Cas 2 : à l'issue de la première itération, au plus un sommet est coloré (lignes 14 - 19). Le spoiler révèle alors des ensembles de $\lfloor\sqrt{n}\rfloor$ sommets reliés entre eux et aux sommets colorés, donc après k itérations, au plus k sommets peuvent être colorés (lignes 14 - 17). Si à l'issue de k itérations, il reste des couleurs disponibles, alors elles ne pourront servir que pour colorer un sommet (chacune) révélé à la dernière itération (lignes 18 - 19). Il s'ensuit que $A(G) \leq k$. Et puisqu'il est toujours possible de colorer au moins $\lfloor\sqrt{n}\rfloor$ sommets (par exemple, ceux de la première itération),

$$\frac{A(G)}{\alpha_k(G)} \leq \frac{k}{\sqrt{n}-1}.$$

□

Il est possible de généraliser ces résultats sur des configurations moins contraignantes. Par exemple, nous avons vu au Chapitre 2 qu'il peut être pertinent d'autoriser la coloration de certains sommets révélés dans le passé, moyennant une pénalité. Le modèle proposé dans [91] consiste à diviser la valeur d'un sommet par un réel $p > 1$ à chaque itération, de sorte qu'un sommet apparu à l'itération i et coloré à l'itération $j > i$ ne comptabilise que pour p^{i-j} unités. Par exemple, si $p = 2$, un sommet coloré une itération après son apparition ne compte que pour moitié. La proposition suivante prend en compte cette possibilité, pour les cas où un unique sommet est révélé à chaque itération.

Proposition 21. *Soit p un réel strictement plus grand que 1. Si un seul sommet est révélé à chaque itération, alors aucun algorithme ne peut garantir un ratio strictement plus grand que*

$$\max\left\{\frac{k+\frac{n-k}{p}}{n}, \frac{k+\frac{n-k-1}{p}}{n-1}\right\}.$$

Démonstration. L'algorithme spoiler est identique que celui de la Proposition 19, mais ne relie les nouveaux sommets qu'à ceux qui ont été colorés à la même itération que leur apparition. L'algorithme on-line A a alors deux possibilités pour colorer un maximum de sommets :

– Soit il colore k sommets à l'itération où ils sont révélés, et exactement $n-k$ à l'itération qui suit leur apparition. Ces $n-k$ sommets supplémentaires ne comptent donc que pour $p^{-1}(n-k)$, soit un total de $k+\frac{n-k}{p}$ sommets colorés. Mais si l'on peut colorer k sommets à l'itération où ils sont révélés et $n-k$ tardivement, c'est que $\alpha_k(G) = n$, d'où

$$\frac{A}{\alpha_k(G)} = \frac{k+\frac{n-k}{p}}{n};$$

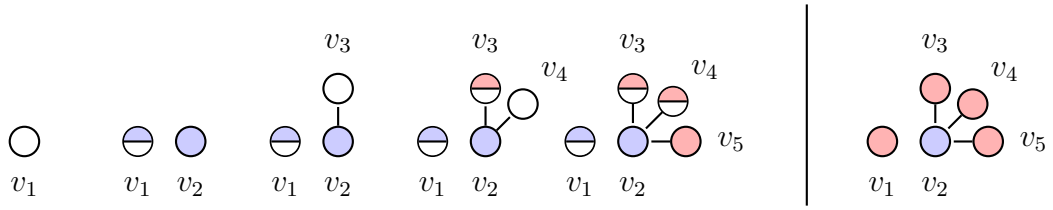
– Soit il colore k sommets à l'itération où ils sont révélés, et exactement $n-k-1$ à l'itération qui suit leur apparition, qui ne comptent donc que pour $p^{-1}(n-k-1)$. Dans ce cas, en prenant la coloration donnée par A excepté le dernier sommet coloré que l'on utilise pour colorer tous les autres sommets non colorés, on obtient $\alpha_k(G) = n-1$, soit

$$\frac{A}{\alpha_k(G)} = \frac{k + \frac{n-k-1}{p}}{n-1}.$$

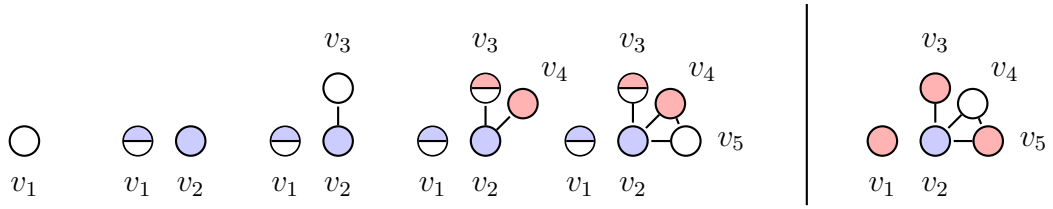
Un exemple est illustré à la Figure 5.3 avec $k = p = 2$ et $n = 5$. □

Remarquons que si p tend vers ∞, on retrouve (asymptotiquement) la garantie de la Propositon 19, où un sommet ne peut être coloré tardivement. Inversement, si $p = 1$ (si l'attente n'est pas pénalisée), on sort de la configuration on-line et le ratio est égal à 1.

Ces garanties sont assez décevantes, mais reposent sur des instances pathologiques qui ne sont pas représentatives du cas général. Il n'est donc pas inutile de développer des heuristiques qui visent à être performantes en moyenne. Tel est l'objet de la section suivante.



$A(G) = 2 + \frac{3}{2}$ (à gauche), $\alpha_2(G) = 5$ (à droite)



$A(G) = 2 + \frac{2}{2}$ (à gauche), $\alpha_2(G) = 4$ (à droite)

Figure 5.3 Un exemple avec $k = p = 2$ et $n = 5$, $\frac{A}{\alpha_k(G)} = \max\left\{\frac{2+\frac{3}{2}}{5}, \frac{2+\frac{2}{2}}{4}\right\} = \max\left\{\frac{7}{10}, \frac{3}{4}\right\} = \frac{3}{4}$

5.2 Heuristiques de coloration on-line

Le problème du plus grand sous-graphe induit k-colorable sur un horizon de temps T est décomposable en $|T|$ sous-problèmes, un par itération. Notons H_t l'ensemble des sommets du graphe de l'itération t, G_t le sous-ensemble de sommets de H_t révélés à l'itération t, et F_{t-1} le sous-ensemble de sommets de H_{t-1} non colorés à l'itération $t-1$. Les décisions étant irrévocables, les sommets de l'ensemble F_{t-1} ne pourront plus être colorés et on peut les enlever du graphe pour les itérations futures. On a donc $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$, et $H_1{=}G_1$ si $H_0 = \varnothing$. On note également $c_t(v)$ la couleur assignée à un sommet v à l'itération $t \in T$.

5.2.1 Les sommets sont révélés un par un

Tout d'abord, considérons le cas où les sommets arrivent un par un (c'est-à-dire $|G_t| = 1$). Lorsqu'un sommet apparaît, toutes ses arêtes incidentes sont également révélées. Celui-ci est soit coloré à l'aide d'une des k couleurs disponibles, soit rejeté. Les décisions sont irrévocables.

De même que dans le Chapitre 4, on fait l'hypothèse que les graphes sont k-colorables, par conséquent on essayera toujours de colorer un sommet révélé. Une seule question se pose donc : quelle couleur doit être attribuée au sommet révélé ? On propose pour cela deux stratégies : First-Fit, qui donne la plus petite couleur possible, et Next-Fit, qui garde en mémoire la dernière couleur utilisée c, et qui assigne la première couleur disponible en considérant la séquence de couleurs $(c{+}1, \cdots, k, 1, \cdots, c)$. First-Fit est la stratégie gloutonne la plus naturelle qui a tendance a saturer séquentiellement les classes de couleur. De cette façon, deux sommets qui ne sont pas adjacents sont de bons candidats pour partager une couleur, laissant ainsi plus de couleurs disponibles pour les sommets suivants. Cependant, cela a tendance à créer des classes de couleurs déséquilibrées, les couleurs de plus petit indice étant sollicitées en priorité. D'où l'intérêt de la politique Next-Fit, qui essaye de répartir au maximum les sommets dans différentes classes de couleur.

Rappelons que Next-Fit est $(2\sqrt{3}\text{-}3)$-compétitif pour la k-coloration on-line d'arêtes [92]. Nous avons également vu à la Section 2.8.1 que colorer un chemin à m arêtes est équivalent à colorer les m sommets de son graphe de ligne, qui est également un chemin. Sur les chemins, First-Fit est $\frac{2}{3}$-compétitif, et Next-Fit $\frac{1}{2}$-compétitif (aucun algorithme déterministe ne peut garantir mieux [92]). Ces garanties soulignent l'interêt que peuvent avoir ces politiques de priorité (du moins sur ces cas particuliers) et motivent leur utilisation ici.

L'algorithme, exécuté à chaque étape de l'horizon de temps, est résumé dans le Tableau 5.3. Il est clair qu'il garantit de colorer k sommets puisqu'au moins k couleurs sont assignées (on ne peut pas espérer faire mieux sur des graphes quelconques d'après la section précédente).

Notons par contre qu'avec la politique Next-Fit, l'ordre du sous-graphe induit par la k-coloration n'augmente pas forcément avec k. Pire : l'ordre peut même diminuer, tel qu'illustré dans l'exemple de la Figure 5.4. L'indice des sommets indique l'itération à laquelle ils sont révélés. La 2-coloration est optimale (tous les sommets sont colorés), mais avec une couleur supplémentaire, un sommet n'est pas coloré.

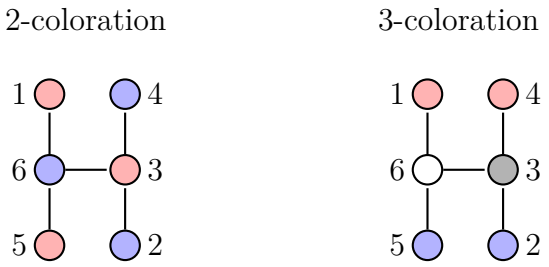


Figure 5.4 2 et 3-colorations obtenues avec la politique Next-Fit

Tableau 5.3 Heuristique on-line gloutonne

Algorithme 15: Glouton(Ω), $\Omega \in$ {First-Fit,Next-Fit}

ENTRÉE : un graphe partiellement coloré, de sommets $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$;
Soit v l'unique sommet de G_t ;
$F_t = \varnothing$;
COLORATION :
si *au moins une couleur parmi* $\{1, \cdots, k\}$ *est disponible pour* v **alors**
    En assigner une à l'aide d'une des deux stratégies Ω :
    $\Omega : \begin{cases} \text{First-Fit} ; \\ \text{Next-Fit} ; \end{cases}$
sinon
    $F_t \leftarrow F_t \cup \{v\}$;

5.2.2 Les sommets sont révélés par paquets

Si les sommets arrivent par paquets (c'est-à-dire $|G_t| > 1$), une question supplémentaire se pose : dans quel ordre les nouveaux sommets devraient-ils être considérés ? C'est ici que l'on fait le lien avec le Chapitre 4 : l'ordre peut être déterminé avec les critères des algorithmes WP1 ou DSAT1, exécutés avec $\theta = 0$ et $W = \omega = 1$. Appelons ces cas particuliers k-LF et k-SLF, respectivement. Quant à elle, la couleur est assignée avec les règles de priorité First-Fit ou Next-Fit.

Puisque les paquets contiennent plusieurs sommets, il est possible d'envisager une autre règle de priorité introduite dans [111], appelée Best-Fit. Celle-ci consiste à assigner à un sommet $v \in G_t$ la couleur qui diminue le degré de saturation pour un minimum de sommets non colorés de G_t, autrement dit celle qui n'est pas disponible pour un maximum de sommets de l'ensemble $G_t \cap N_v$. Les cas d'égalité sont départagés par la couleur de plus petit indice. Par exemple, à la Figure 5.5, si l'on cherche à colorer le sommet v, la politique Best-Fit sélectionnerait la couleur rouge, car elle est déjà non disponible pour tous les voisins de v, alors que la couleur bleue n'est pas disponible que pour deux voisins, et la couleur grise que pour un seul.

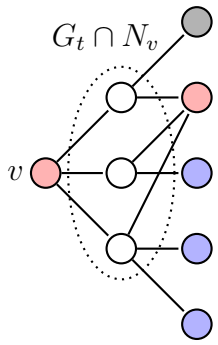


Figure 5.5 Politique Best-Fit, un exemple

Notons que la procédure Best-Fit et l'algorithme RLF sont similaires dans la mesure où les deux stratégies essayent de regrouper des sommets ayant un maximum de voisins en commun. De plus, si G_t contient un seul sommet v, alors $G_t \cap N_v = \varnothing$. Best-Fit choisit alors simplement la couleur de plus petit indice et est identique à First-Fit. Les algorithmes k-LF et k-SLF incorporant l'une de ces trois politiques sont résumés dans les Tableaux 5.4 et 5.5.

Tableau 5.4 Algorithme on-line k-LF

Algorithme 16: k-LF(Ω), $\Omega \in$ {First-Fit,Next-Fit,Best-Fit}

ENTRÉE : un graphe partiellement coloré, de sommets $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$;
INITIALISATION :
$F_t = \varnothing$;
Ordonner les sommets de G_t par ordre de degré non croissant dans une liste $(v_1, \cdots, v_r)$;
COLORATION :
pour $v = v_1, \cdots, v_r$ **faire**
    si *au moins une couleur parmi* $\{1, \cdots, k\}$ *est disponible pour* v **alors**
        En assigner une à l'aide d'une des trois stratégies Ω :
        Ω : First-Fit ; / Next-Fit ; / Best-Fit ;
    sinon
        $F_t \leftarrow F_t \cup \{v\}$;

Tableau 5.5 Algorithme on-line k-SLF

Algorithme 17: k-SLF(Ω), $\Omega \in$ {First-Fit,Next-Fit,Best-Fit}

ENTRÉE : un graphe partiellement coloré, de sommets $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$;
INITIALISATION :
$F_t = \varnothing$;
Déterminer l'ensemble $A(v)$ des couleurs non disponibles pour tout $v \in G_t$;
COLORATION :
tant que $G_t \neq \varnothing$ **faire**
    Choisir un sommet $v \in G_t$ qui maximise $|A(v)|$;
    (les cas d'égalité son départagés par le plus grand degré)
    Assigner une couleur à v à l'aide d'une des trois stratégies Ω :
    Ω : First-Fit ; / Next-Fit ; / Best-Fit ;
    Retirer v de G_t, mettre à jour $A(u)$ pour tout $u \in G_t \cap N_v$ et transférer ceux pour lesquels $A(u) = k$ vers F_t ;

5.3 Relaxation

Idéalement, l'optimisation de l'allocation de ressources dans un réseau de téléphonie mobile sur un horizon de temps devrait considérer deux objectifs simultanément : maximiser le nombre d'usagers connectés au réseau, et minimiser les changements d'affectation au cours du temps, appelés *transferts inter/intra-cellulaires* (ou *handovers*, en anglais) [112, 113]. Il est possible qu'un usager connecté au réseau change de canal au cours du transfert de données numériques (typiquement lors d'un changement de cellule dû à un déplacement), mais cela augmente la consommation d'énergie ainsi que la probabilité de coupure [113]. Ces changements d'assignation doivent donc être évités si possible.

Ceci motive la relaxation suivante du problème de coloration on-line : les assignations ne sont plus complètement irrévocables, en ce sens que les sommets colorés doivent rester colorés, mais la couleur peut changer, bien que cela devrait être évité. Notons qu'une telle relaxation n'améliore pas les meilleures garanties possibles. La difficulté réside ici dans le fait que les deux objectifs sont conflictuels : s'il est plus important de maximiser le nombre de sommets colorés, il semble plus efficace de réinitialiser toutes les classes de couleur et de relancer les algorithmes d'affectation dès l'arrivée d'un ou plusieurs sommets, quitte à ce que tous les sommets changent de couleur. En revanche, si plus d'importance est donnée à la minimisation du nombre de changements de canaux au cours du temps, alors il est peut-être plus intéressant de laisser les classes de couleur telles quelles, et d'insérer les nouveaux sommets si possible.

Lorsque la saturation est proche, il est probable que le graphe soit fortement contraint par de mauvaises décisions prises dans le passé et qu'il soit impossible de colorer tous les sommets. Le cas échéant, nous proposons deux approches pour essayer de « réparer » une telle coloration :

- les classes de couleurs sont réinitialisées et reconstruites suivant les règles de k-LF ou k-SLF. Afin de limiter les changements de couleurs, un sommet qui était coloré auparavant récupère sa couleur lorsque cela est possible. Cette première stratégie s'inscrit dans la famille des algorithmes de *réparation* [114] mentionnés à la Section 2.8.2, et est couramment utilisée dans la littérature comme référence de comparaison pour les algorithmes d'allocation de ressources [111].
- les classes de couleurs restent telles quelles et l'on essaye de colorer les sommets rejetés par un procédé de recherche locale tout en pénalisant les changements de couleurs. Cette stratégie est inspirée de la Recherche Tabou appelée TABUCOL présentée au Chapitre 2, dont l'efficacité pour la coloration classique est incontestable [83].

L'algorithme incorporant l'une ou l'autre de ces stratégies et résumé au Tableau 5.6, et est exécuté à chaque étape de l'horizon de temps. On dit qu'une amélioration a lieu si le nombre total de sommets colorés a augmenté (quelque soit le nombre de changements de couleurs), et que tous les sommets colorés ont conservé une couleur au cours du temps. Les changements issus de ces procédures sont pris en compte uniquement s'ils donnent lieu à une amélioration de la solution. Dans les deux sections qui suivent, nous donnons des précisions sur ces deux stratégies de réparation et de recherche locale.

Tableau 5.6 Algorithme on-line permettant des changements de couleurs

Algorithme 18: k-(Ω, Φ, Ψ), $\Omega \in$ {First-Fit,Next-Fit,Best-Fit}, $\Phi \in$ {LF,SLF}, $\Psi \in$ {RÉPARER,TABU}

ENTRÉE : un graphe partiellement coloré, de sommets $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$;
$F_t = \varnothing$;
COLORATION :
Assigner une couleur aux sommets de G_t à l'aide d'une des six stratégies (Φ, Ω) :
$\Phi : \begin{cases} k\text{-LF}(\Omega) ; \\ k\text{-SLF}(\Omega) ; \end{cases}$ $\Omega : \begin{cases} \text{First-Fit} ; \\ \text{Next-Fit} ; \\ \text{Best-Fit} ; \end{cases}$
si *un sommet v est rejeté* **alors**
    $F_t \leftarrow F_t \cup \{v\}$;
si $F_t \neq \varnothing$ **alors**
    Reconsidérer les affectations avec l'une des deux stratégies Ψ :
    $\Psi : \begin{cases} \text{RÉPARER}(\Phi, \Omega); \\ \text{TABU}; \end{cases}$
    si *aucune amélioration n'a lieu* **alors**
        Annuler les modifications issues de Ψ ;

5.3.1 Algorithme de réparation

L'algorithme de réparation, résumé dans le Tableau 5.7, peut être décrit comme suit. Les classes de couleurs sont reconstruites à l'aide des algorithmes k-LF ou k-SLF, munis des règles de priorité supplémentaires suivantes. Lorsqu'un sommet $v \in H_t$ est traité, deux cas de figure se présentent :

– si $v \in H_{t-1}$, c'est que v possédait la couleur $c_{t-1}(v)$ à l'itération précédente. L'algorithme essaye alors de lui attribuer en priorité cette couleur ;
– si cette couleur n'est plus disponible, ou si $v \in G_t$, une couleur est attribuée à v suivant les politiques First-Fit, Next-Fit ou Best-Fit.

Cette stratégie présente au moins deux défauts. Tout d'abord, il peut arriver que certaines classes de couleurs contiennent les mêmes sommets avant et après la procédure de réparation, mais que les indices des classes aient été interchangés. Il est alors possible de minimiser ces changements inutiles au moyen d'une simple permutation des couleurs, décrite dans le paragraphe suivant.

Supposons qu'à l'itération $t+1$, k couleurs soient utilisées. Considérons donc le graphe biparti complet $K_{kk} = (U_1 \cup U_2, E)$, où toute arête $\{i, j\} \in E$ est pondérée par un poids c_{ij} égal au nombre de sommets de l'itération t qui avaient la couleur i, parmi ceux de l'itération $t+1$ qui ont la couleur j. En notant V_i^t l'ensemble des sommets de couleur i à l'itération t, on a

$$c_{ij} = |V_i^t \cap V_j^{t+1}|.$$

Le couplage de poids maximum donne alors la permutation des couleurs σ qui maximise le nombre de sommets qui récupèrent leur couleur initiale. Plus précisément, pour les sommets de H_t, la couleur $\ell_2 \in U_2$ est remplacée par la couleur $\ell_1 \in U_1$ du couplage obtenu, et le graphe résultant est noté $\sigma(H_t)$. Un tel couplage peut être calculé en un temps $\mathcal{O}\left(|E|^2\sqrt{|U|}\log C\right) = \mathcal{O}\left(k^{9/2}\log C\right)$ [115], où C représente le poids maximum sur les arêtes de E, et $U = U_1 \cup U_2$. Un exemple est illustré à la Figure 5.6. Supposons que l'on dispose de $k = 3$ couleurs disponibles. À l'itération t, les quatre sommets a, b, c, d de H_t sont colorés. Une itération après, le sommet e apparaît mais aucune couleur ne lui convient. On réinitialise donc les classes de couleur et l'on parvient à colorer le graphe entièrement, mais au prix de quatre changements de couleur (a, b, c et d ont perdu leur couleur initiale). Si la couleur verte devient bleue, aucun sommet ne récupère sa couleur initiale, donc le poids sur l'arête correspondante du graphe K_{33} est nul. Par contre, si la couleur verte devient rouge, le sommet a récupère sa couleur initiale, donc le poids correspondant est égal à 1. On pondère d'une manière similaire les autres arêtes du graphe, et l'on obtient un couplage optimal de poids égal à 3. Celui-ci permet aux sommets a, b, c de récupérer leur couleur initiale. On parvient ainsi à colorer les quatre sommets a, b, c et d, au prix d'un unique changement de couleur (celui de d).

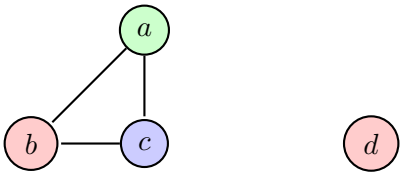
Deuxièmement, il est impossible de garantir que tous les sommets de l'ensemble H_{t-1} soient bel et bien colorés à l'issue de la procédure de réparation. Un exemple est illustré à la Figure 5.7, avec $k = 1$ couleur disponible. À la première itération, un sommet isolé apparaît et est donc coloré. À l'itération suivante, 4 sommets sont révélés, et un seul parmi ces 4 nouveaux

sommets peut-être coloré. Puisque 3 sommets sont rejetés, la procédure de réparation est exécutée et trouve un ensemble indépendant de 3 sommets. Cependant, cet ensemble indépendant ne contient pas le sommet de la première itération et on ne peut donc pas tenir compte de cette réparation, malgré le fait que le nombre de sommets colorés a augmenté. Pour minimiser la probabilité que ce scénario se présente, les cas d'égalité sont départagés en donnant la priorité aux sommets de H_{t-1} au cours de la procédure de réparation.

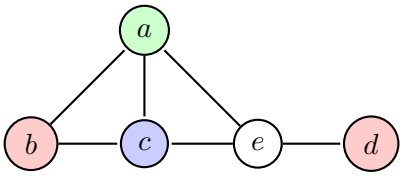
Tableau 5.7 Algorithme de réparation

Algorithme 19: RÉPARER(Φ, Ω), $\Phi \in$ {LF,SLF}, $\Omega \in$ {First-Fit,Next-Fit,Best-Fit}

ENTRÉE : un graphe partiellement coloré $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$;
$F_t = \varnothing$;
$V_i = \varnothing$ pour toute couleur $i = 1, \cdots, k$;
RÉPARATION :
Assigner une couleur aux sommets de H_t à l'aide d'une des deux stratégies Φ :
$\Phi : \begin{cases} k\text{-LF} ; \\ k\text{-SLF} ; \end{cases}$
Lorsqu'un sommet v est traité, utiliser les règles de priorité suivantes ;
si $v \in H_{t-1}$ **alors**
    Essayer de lui affecter la couleur $c_{t-1}(v)$;
si *ce n'est pas possible ou* ***si*** $v \in G_t$ **alors**
    si *au moins une couleur parmi* $\{1, \cdots, k\}$ *est disponible pour* v **alors**
        En assigner une à l'aide d'une des trois stratégies Ω :
        $\Omega : \begin{cases} \text{First Fit} ; \\ \text{Best Fit} ; \\ \text{Next Fit} ; \end{cases}$
    sinon
        $F_t \leftarrow F_t \cup \{v\}$;
Calculer σ et mettre les classes de couleur à jour en conséquence ;

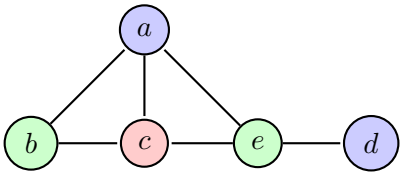


$H_t = \{a, b, c, d\}$



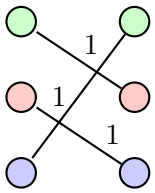
$H_{t+1} = H_t \cup G_{t+1} = \{a, b, c, d\} \cup \{e\}$

Réparation

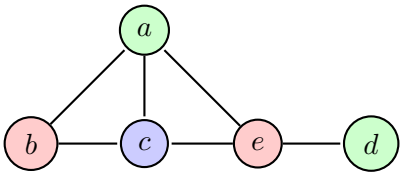


H_{t+1} après réparation

σ



Couplage de poids maximum



$\sigma(H_{t+1})$

Figure 5.6 Un couplage de poids maximum pour diminuer des changements de couleurs inutiles

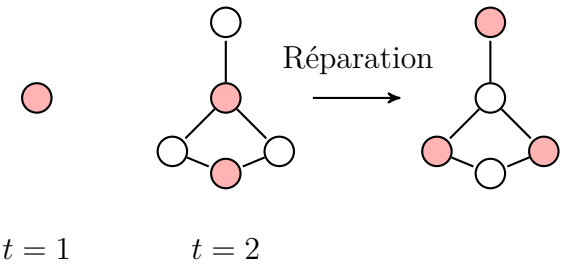


Figure 5.7 Un sommet de H_1 n'est plus coloré à l'issue de la réparation de H_2

5.3.2 Algorithme de recherche locale

Dans la seconde stratégie que avons développée, les sommets qui ont été rejetés sont placés dans une file et sont traités séquentiellement par une procédure de recherche locale qui tente de les insérer en modifiant localement la coloration tout en pénalisant les changements de couleurs. Tel qu'expliqué au Chapitre 2, parcourir l'espace de recherche des solutions (non nécessairement réalisables) de façon efficace peut être réalisé au moyen d'une Recherche Tabou [82] dont les ingrédients principaux sont décrits ci-dessous.

Solution initiale. Les sommets rejetés sont placés dans une file, le premier sommet étant celui qui a été rejeté en premier. Les sommets de $H_t \setminus G_t$ conservent leur couleur, ainsi que ceux de G_t qui ont pu en obtenir une. Le premier sommet de la file prend une couleur aléatoire parmi l'ensemble $\{1, \cdots, k\}$, les autres sommets de la file ne sont pas considérés et demeurent non colorés.

Fonction objectif. Typiquement, cette solution initiale n'est pas réalisable. En notant $E(H_t)$ l'ensemble des arêtes du graphe de sommets H_t, il existe au moins une arête de $E(H_t)$ dont les extrémités sont colorées avec la même couleur. La fonction objectif qui est minimisée est le nombre de tels conflits. Lorsque celui-ci est nul, la solution est réalisable et l'on pourrait arrêter le processus de recherche. Comme on cherche également à minimiser le nombre de changements de couleur, on ajoute un second terme à la fonction objectif de façon à pénaliser ces changements. Chaque changement augmente la fonction objectif de p unités, où p est un paramètre strictement positif. L'expression explicite de la fonction objectif peut s'écrire comme suit :

$$f(H_t, p) = |\{\{u, v\} \in E(H_t) \mid c_t(u) = c_t(v)\}| + p \sum_{v \in H_t \setminus G_t} \min\{1, |c_t(v) - c_{t-1}(v)|\}.$$

Le paramètre p a été fixé a 0.1 dans les tests numériques. Ceci signifie que la fonction objectif ne s'améliore pas si 10 changements de couleurs ont été effectués pour éliminer un conflit.

Voisinage. La Recherche Tabou explore l'espace de recherche en modifiant la couleur des sommets conflictuels, et se déplace vers la meilleure solution trouvée, quitte à dégrader la fonction objectif courante. Le graphe obtenu après avoir attribué la couleur ℓ_2 à un sommet v de couleur ℓ_1 est noté $H_t \oplus (v, \ell_2)$, et l'on appelle *voisinage* de la solution courante l'ensemble

$$\mathcal{N}(v) = N_v \cap \{u \in H_t \mid c_t(u) = \ell_1\},$$

autrement dit l'ensemble des sommets adjacents à v, de même couleur que v.

Liste taboue. Pour ne pas rester bloqué dans un minimum local, le statut *tabou* est donné au mouvement inverse qui a été réalisé. Par exemple, lorsqu'un sommet v de couleur ℓ_1 reçoit la couleur ℓ_2, le mouvement inverse, (v, ℓ_1), est placé dans une liste dite taboue notée $\mathcal{T}$ pendant un nombre donné d'itérations. Il faut que ce nombre soit suffisamment grand pour éviter de retourner vers un minimum local, mais pas trop pour ne pas restreindre l'espace de recherche inutilement. Nous l'avons fixé à $\sqrt{|\mathcal{N}(v)|}$. Tout mouvement appartenant à $\mathcal{T}$ ne peut pas être effectué, sauf s'il satisfait le *critère d'aspiration*.

Critère d'aspiration. Si une solution trouvée s'avère meilleure que la meilleure solution enregistrée depuis le début de la recherche, mais que celle-ci nécessite un mouvement tabou, alors un critère dit d'aspiration passe outre ce statut et permet sa réalisation. Ce mécanisme assure que de bonnes solutions potentielles ne soient pas exclues de l'espace de recherche.

Critère d'arrêt. La recherche est interrompue après 10 itérations sans amélioration de la fonction objectif.

Pseudocode. Les ingrédients de cette Recherche Tabou étant définis, ses étapes sont résumées dans le Tableau 5.8.

Il peut être tentant de vouloir effectuer cette recherche sur l'ensemble des sommets rejetés simultanément (comme dans le TABUCOL original), mais si aucune solution réalisable n'est trouvée, ceci est problématique car aucun sommet ne peut être coloré. Décolorer les sommets conflictuels peut être synonyme de décolorer des sommets de H_{t-1}, ce qui n'est pas permis.

Tableau 5.8 Recherche Tabou

Algorithme 20: TABOU

ENTRÉE : une liste $(u_1, \cdots, u_r)$ de sommets rejetés ;
pour $u = u_1, \cdots, u_r$ **faire**
    INITIALISATION :
    Donner une couleur parmi $\{1, \cdots, k\}$ à u de façon aléatoire ;
    Soit $f^*(H_t, p)$ le nombre de conflits ;
    Poser $\mathcal{T} = \varnothing$;
    RECHERCHE TABOU :
    tant que *un critère d'arrêt n'est pas satisfait* **faire**
        Déterminer le meilleur mouvement (v, ℓ) qui ne soit pas dans $\mathcal{T}$ ou qui diminue $f^*(H_t, p)$;
        Mettre à jour la liste $\mathcal{T}$;
        Assigner la couleur ℓ au sommet v ;
        si $f(H_t \oplus (v, \ell), p) < f^*(H_t, p)$ **alors**
            $f^*(H_t, p) \leftarrow f(H_t \oplus (v, \ell), p)$;
        $H_t \leftarrow H_t \oplus (v, \ell)$;
    si *u n'a pas de couleur* **alors**
        $F_t \leftarrow F_t \cup \{u\}$;

5.4 Résultats numériques

Tel que mentionné en introduction du Chapitre 4, les paramètres $\alpha_k(G)$ et $\chi(G)$ sont reliés par la relation

$$\chi(G) = \min\{k \in \mathbb{N} \mid \alpha_k(G) = n\},$$

par conséquent la performance de nos algorithmes peut être évaluée en se référant à $\chi(G)$. Nous avons donc testé nos heuristiques sur les instances DIMACS [2] pour lesquelles le paramètre $\chi(G)$ est connu. Ces instances servent de référence dans la littérature pour évaluer la performance des algorithmes de coloration, et présentent les caractéristiques suivantes : $11 \leq n \leq 559$, $20 \leq m \leq 18\,707$ et $4 \leq \chi \leq 73$. Dans le cadre de nos expériences, nous avons supposé que les sommets (avec l'ensemble de leurs arêtes incidentes) sont révélés dans un ordre aléatoire.

2. `http://mat.gsia.cmu.edu/COLOR/instances.html`

5.4.1 Performances pour $k \geq \chi$

Les sommets apparaissent un par un

Les résultats numériques sont présentés dans le Tableau 5.11 ; pour chaque instance, nous avons exécuté l'algorithme glouton avec les politiques First-Fit et Next-Fit pour les valeurs de k appartenant à l'ensemble $\{\chi, \chi+1, \cdots, \chi+10\}$. Une entrée du tableau correspond au nombre de sommets colorés (colonne 'col.') ainsi qu'au temps de calcul CPU en secondes (colonne 'Δt'), arrondi au dixième. Lorsque la solution est optimale, elle est écrite en gras de façon à faciliter la lecture du tableau. Une synthèse visuelle des résultats est représentée dans les graphiques de la Figure 5.8. Pour chaque valeur de $\{\chi, \chi+1, \cdots, \chi+10\}$ est donnée le pourcentage d'instances pour lesquelles tous les sommets ont été colorés. On observe que la politique First-Fit permet d'atteindre la solution optimale pour 42% des instances, alors que la politique Next-Fit ne l'atteint que dans 10% des cas. Tel que mentionné plus haut, avec la politique Next-Fit, ajouter une couleur peut même dégrader la solution, ce qui explique pourquoi la courbe associée à Next-Fit n'est pas croissante avec k. Au global, la stratégie First-Fit est plus efficace que Next-Fit, car la première courbe est toujours située au-dessus de la seconde. Cependant, il est intéressant de noter que si l'on regarde les mêmes données en acceptant des résultats avec une erreur relative inférieure à 10% (ce qui signifie que l'on tolère les solutions où au moins 90% des sommets sont colorés), alors la politique Next-Fit devient plus performante que First-Fit, avec environ 72% d'instances acceptées, soit près de 7% de plus que First-Fit. Ceci illustre le fait que Next-Fit n'atteint pas souvent la solution optimale, mais a tendance à colorer de nombreux sommets quelque soit le graphe considéré. En revanche, First-Fit trouve plus souvent la solution optimale, mais échoue complètement sur certaines instances telles que les graphes `queen6-6` - `queen11-11`.

Les sommets apparaissent par paquets

Commençons par analyser l'autre cas extrême, lorsque tous les sommets apparaissent en un seul paquet. Autrement dit une seule itération a lieu, et le scénario est off-line. Les résultats sont présentés dans le Tableau 5.12, ou sous forme de graphique à la Figure 5.9. On observe que la stratégie k-SLF avec la politique Best-Fit est la plus efficace, avec 68% d'instances résolues à l'optimalité. La courbe correspondante est toujours au dessus des autres, et avec $k = \chi + 9$ couleurs, tous les sommets de toutes les instances sont colorés. Tel que mentionné plus haut, ceci n'est pas surprenant, car cette stratégie est similaire à celle de de l'algorithme RLF, qui s'est avéré expérimentalement plus performant que LF ou SLF d'après de nombreux tests numériques.

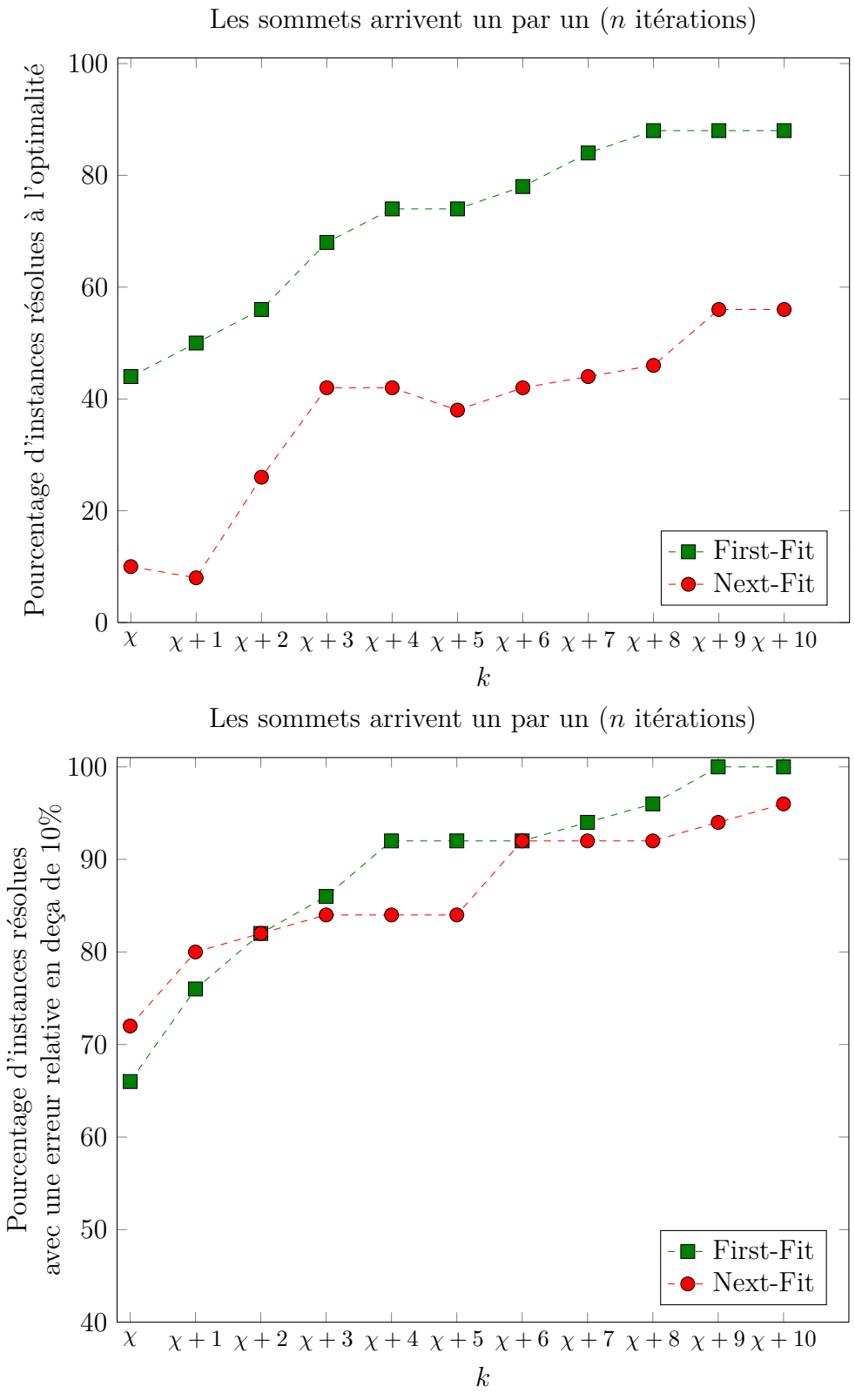


Figure 5.8 Graphiques associés au Tableau 5.11

Notons également que les courbes de k-SLF dominent toujours celles de k-LF, ce qui signifie que l'algorithme k-SLF est toujours plus performant que k-LF, et ceci quelque soit la politique de priorité considérée. De plus, la politique Next-Fit a tendance à dégrader les performances de k-SLF, puisque seulement 43% des instances sont résolues à l'optimalité avec cette stratégie. Avec $k = \chi$ couleurs, k-LF couplé à la politique Best-Fit est presque aussi compétitif que k-SLF couplé à First-Fit ou Best-Fit, avec environ 65% d'instances résolues de façon optimale. Au global, ceci suggère que la stratégie Best-Fit est la stratégie gagnante, et Next-Fit celle à éviter.

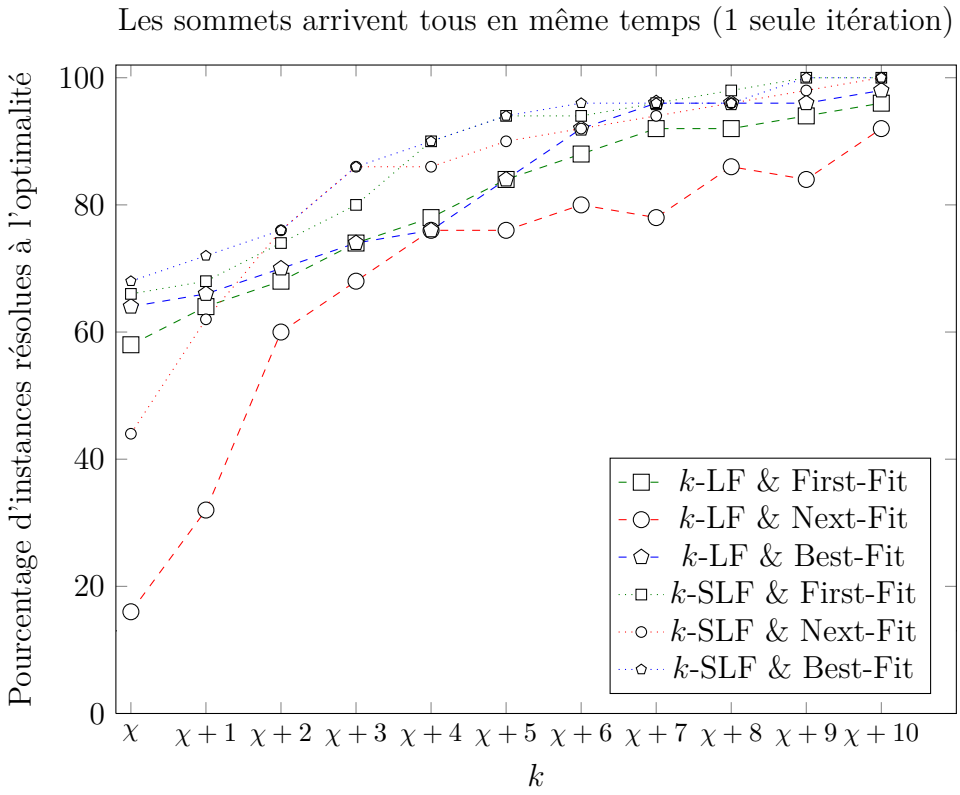


Figure 5.9 Graphique associé au Tableau 5.12

5.4.2 Écart avec la solution optimale

Pour analyser les résultats pour des paquets de taille intermédiaire, nous considérons les mesures de performances suivantes pour un ensemble d'instances S : *l'écart absolu total*

(TAPD) et *l'écart moyen relatif* (ARPD [3]) qui mesurent l'erreur entre la solution optimale d'une instance (le nombre de sommets n_s) avec la solution a_s obtenue par un algorithme. Ces grandeurs, exprimées en pourcentage, sont définies de la façon suivante :

$$\begin{cases} TAPD = 100\ \dfrac{\sum_{s\in S} n_s - a_s}{\sum_{s\in S} n_s} \\ \\ ARPD = 100\ \dfrac{\sum_{s\in S} \frac{n_s - a_s}{n_s}}{|S|} \end{cases}$$

Le TAPD mesure le nombre total de sommets non colorés, mais peut suggérer de fausses conclusions si les graphes ont des tailles très différentes. Par exemple, supposons que S comprenne deux instances s_1 et s_2 avec $n_{s_1} = 10$ et $n_{s_2} = 100$. Si un algorithme A_1 trouve des solutions de taille 9 (pour s_1) et 100 (pour s_2), alors son TAPD vaut $\frac{100}{110} = 0.909$. Si un second algorithme A_2 trouve des solutions $a_{s_1} = 10$ et $a_{s_2} = 90$, son TAPD est 10 fois supérieur à celui de A_1 alors que les deux algorithmes ont résolu une instance de façon optimale, et une autre avec 10% de sommets colorés en moins que la solution optimale. Il est donc hâtif de conclure que A_1 est plus performant que A_2. D'où l'intérêt du ARPD qui corrige ce biais, donnant aux deux algorithmes la même valeur.

Les histogrammes des Figures 5.10 et 5.11 représentent les résultats pour des tailles de paquets égales à 1, $\left\lfloor \frac{n}{2} \right\rfloor$, $\left\lfloor \frac{n}{10} \right\rfloor$, et n. On observe que plus il y a d'itérations (ou plus la taille des paquets est petite), plus il est difficile de colorer tous les sommets. Ceci est vrai pour k-LF et k-SLF, et pour les trois politiques de priorité . Par exemple, le TAPD pour k-LF couplé à Best-Fit augmente de 9.5% (pour n itérations) à 12% (pour 1 itération), et son ARPD augmente de 7.5% à 9%. Les évolutions sont similaires pour les autres stratégies, mais sont moins significatives avec Next-Fit. Le fait que les ARPD et TAPD de k-LF avec Next-Fit ne s'améliorent pas beaucoup peut s'expliquer par le fait que ses performances demeurent moyennes même avec des paquets de taille n (voir la Figure 5.9). Sinon, de façon générale, il semble logique que plus l'horizon de temps est grand, plus l'algorithme est susceptible de prendre de mauvaises décisions et donc moins ses performances moyennes sont bonnes.

Cependant, on observe que lorsque les sommets sont révélés un par un, les TAPD et ARPD ne sont pas systématiquement pires qu'avec des paquets de taille $\left\lfloor \frac{n}{10} \right\rfloor$, ce qui semble contredire le fait que plus l'horizon de temps est grand, moins les performances sont bonnes. Ceci peut s'expliquer par le fait que lorsque la taille des paquets est petite, l'ordre d'arrivée des sommets a un fort impact sur la qualité de la solution. Il existe toujours un ordre statique pour lequel

3. Ces abréviations proviennent de l'anglais : *Total Absolute Percentage Deviation* et *Average Relative Percentage Deviation*.

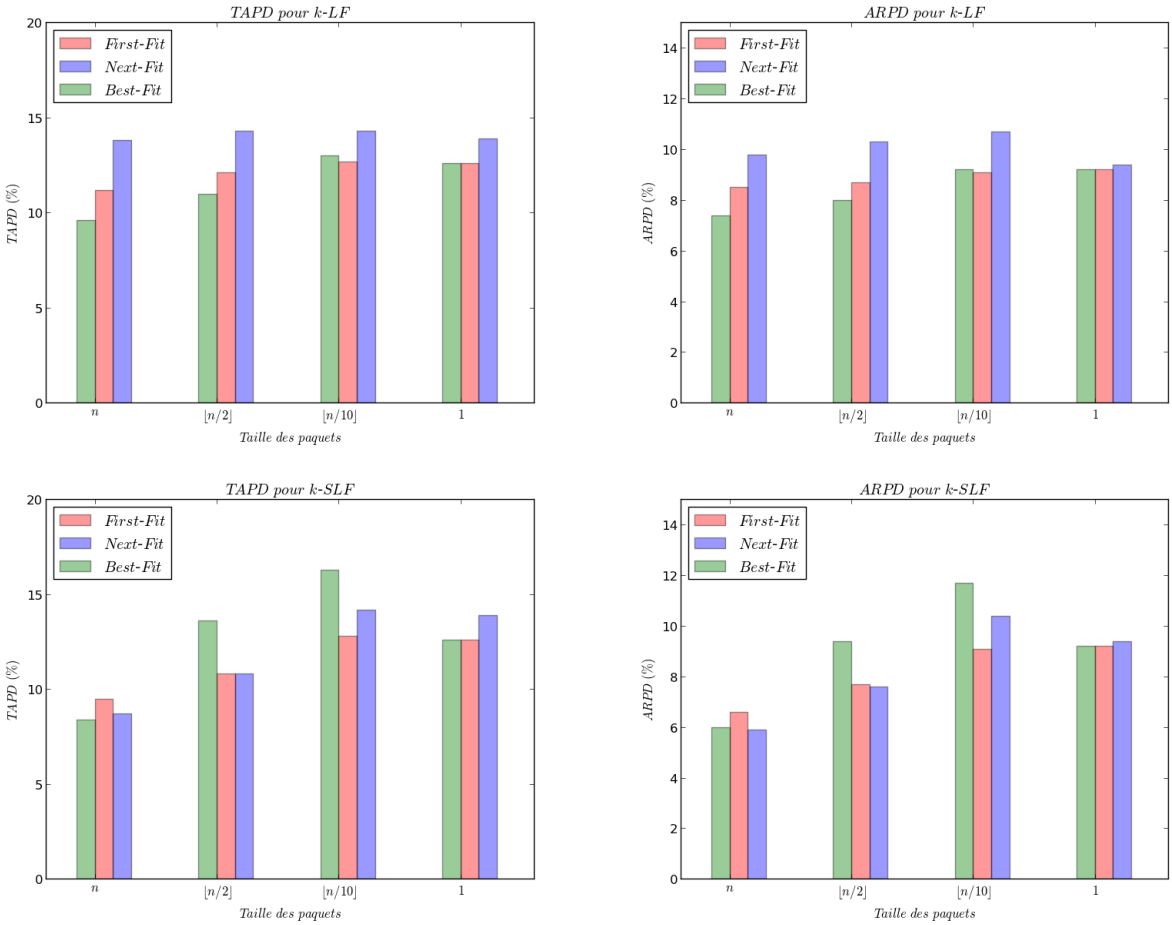


Figure 5.10 Histogrammes représentant les TAPD et ARPD pour k-LF et k-SLF, $k = \chi$

n'importe quel algorithme glouton peut trouver la solution optimale, et inversement, certains ordres peuvent donner lieu à de très mauvaises colorations. Lorsque la taille des paquets est grande, ceci a peu d'importance car l'algorithme peut réarranger l'ordre des sommets, mais lorsque elle est petite, l'ordre d'arrivée des sommets semble être plus déterminant que la politique de choix de la couleur considérée.

Enfin, on observe que pour l'algorithme k-LF, Best-Fit est la stratégie gagnante quel que soit le nombre d'itérations, et que Next-Fit est la stratégie la moins efficace. Pour k-SLF, les choses ne sont pas aussi claires, la stratégie Best-Fit ayant des TAPD et ARPD moins bons pour des tailles intermédiaires. La politique First-Fit est donc à privilégier pour k-SLF.

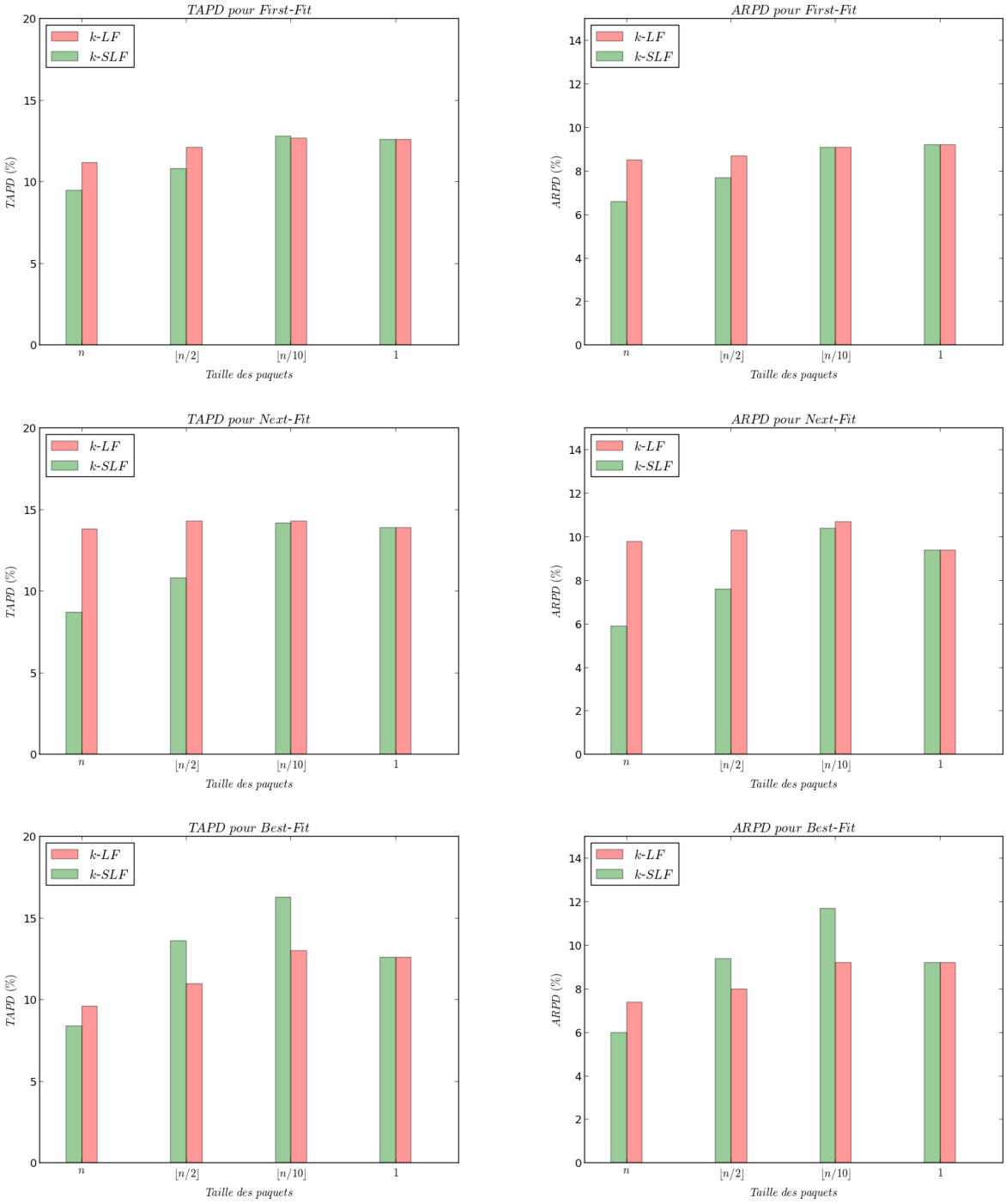


Figure 5.11 Histogrammes des TAPD et ARPD pour les politiques First-Fit, Next-Fit, et Best-Fit, $k = \chi$

5.4.3 Relaxation

Nous poursuivons cette section par une série de tests pour la relaxation du problème où les sommets peuvent changer de couleur au cours du temps. L'algorithme k-LF a été exécuté sur les 50 instances, avec des paquets de taille $\left\lfloor \frac{n}{10} \right\rfloor$, $k = \chi$ couleurs, et avec les stratégies suivantes :

- uniquement k-LF (la relaxation n'est pas considérée) ;
- k-LF et l'algorithme de réparation k-LF sans la permutation σ (k-LF & k-LF) ;
- k-LF et l'algorithme de réparation k-LF avec la permutation σ (k-LF & k-LF & σ) ;
- k-LF et la Recherche Tabou avec comme critère d'arrêt 10 itérations sans amélioration (k-LF & Tabou) ;
- k-LF et la Recherche Tabou avec comme critère d'arrêt 50 itérations sans amélioration (k-LF & Long Tabou).

Les résultats sont présentés dans le Tableau 5.9. Les abréviations 'col.' , 'cha.', et 'Δt' désignent respectivement le nombre de sommets colorés, le nombre de changements de couleurs effectués et le temps de calcul en secondes. Afin de visualiser une éventuelle dominance d'une de ces stratégies sur les autres, les résultats sont représentés dans l'espace des fonctions objectifs à deux dimensions (voir Figure 5.12), avec un axe par objectif : le TAPD/ARPD en ordonnée, et le nombre de changements de couleurs en abscisse. L'algorithme optimal vis-à-vis des deux objectifs produirait des résultats dans le coin inférieur gauche, avec un minimum de sommets non colorés et de changements de couleurs. Ces changements ont été normalisés de la façon suivante : l'algorithme qui en produit le plus génère un point d'abscisse 100 .

On observe que lorsque la relaxation n'est pas considérée, aucun changement de couleur n'a lieu, et 12.8% des sommets ne sont pas colorés. À l'autre extrême, l'algorithme de réparation sans la permutation σ ne colore pas 12.3% des sommets, au prix d'un maximum de changements de couleurs (100, après normalisation). La permutation σ permet de diminuer légèrement les changements de couleurs (90). Entre ces valeurs limites, les stratégies intermédiaires produisent des résultats venant épouser le front de Pareto de l'espace des fonctions objectifs, c'est-à-dire la limite à partir de laquelle toute amélioration du premier objectif dégrade le second, et inversement. L'algorithme de recherche locale avec le critère d'arrêt à 10 itérations sans amélioration colore presque autant de sommets que l'algorithme de réparation (12.5% de sommets non colorés), mais avec uniquement 5 changements de couleurs. Les meilleurs résultats sont obtenus avec cette stratégie exécutée avec un critère d'arrêt à 50 itérations sans améliorations : un total de 12.1% sommets ne sont pas colorés, pour 9 changements de couleur.

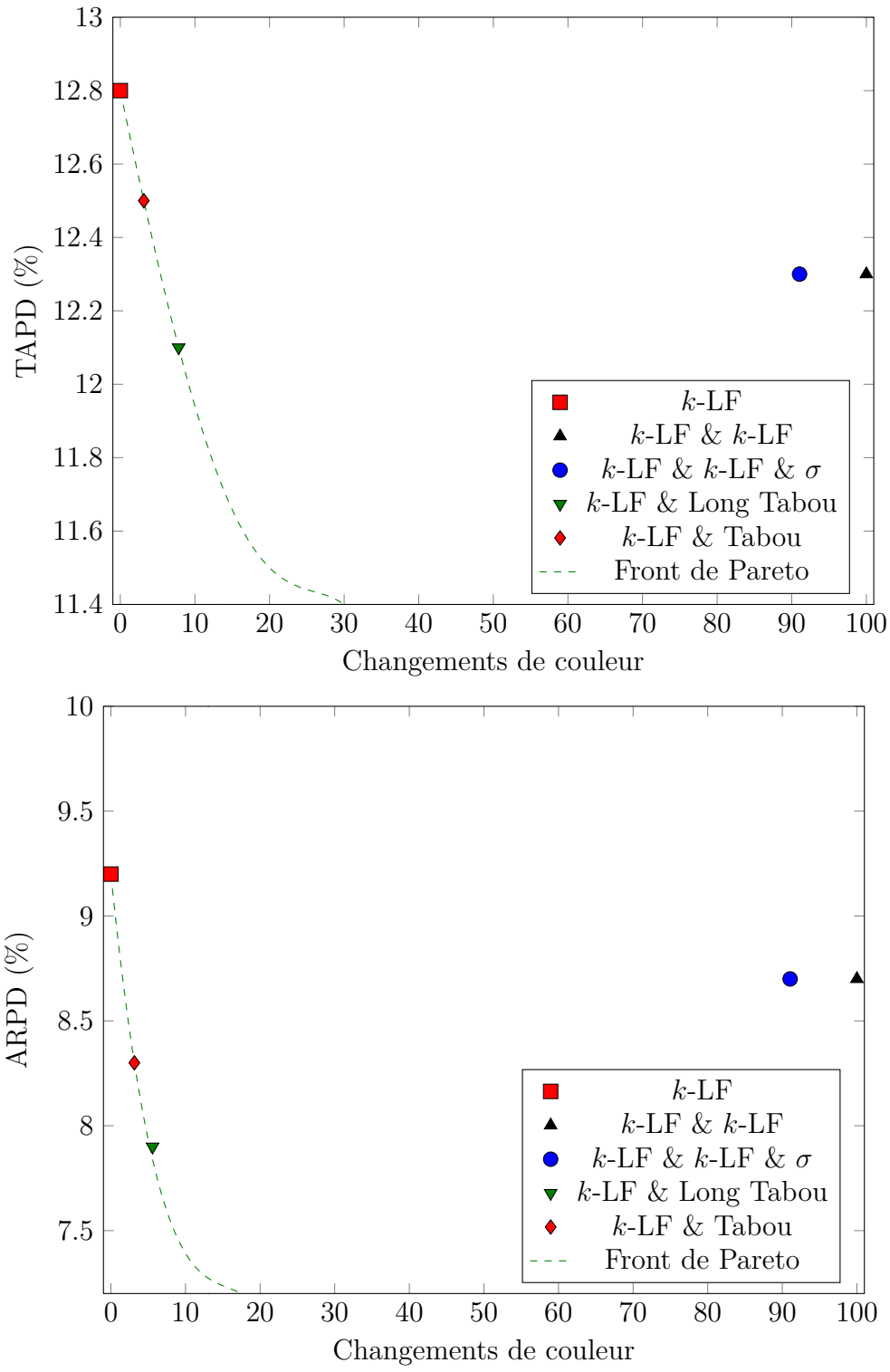


Figure 5.12 Relaxation avec $k = \chi$; représentation des résultats dans l'espace des fonctions objectifs

Tableau 5.9 Relaxation : résultats numériques avec $k = \chi$, $|G_t| = \left\lfloor \frac{n}{10} \right\rfloor$

	k-LF			k-LF & k-LF			k-LF & k-LF & σ			k-LF & Long Tabou			k-LF & Tabou		
INSTANCE (n, χ)	col.	cha.	Δt	col.	cha.	Δt	col.	cha.	Δt	col.	cha.	Δt	col.	cha.	Δt
fpsol2.i.1 (269, 65)	269	0	0.1	269	0	0.1	269	0	0.1	269	0	0.1	269	0	0.1
fpsol2.i.2 (363, 30)	363	0	0.1	363	0	0.1	363	0	0.1	363	0	0.1	363	0	0.1
fpsol2.i.3 (363, 30)	363	0	0.1	363	0	0.1	363	0	0.1	363	0	0.1	363	0	0.1
inithx.i.1 (519, 54)	519	0	0.2	519	0	0.2	519	0	0.2	519	0	0.2	519	0	0.2
inithx.i.2 (558, 31)	558	0	0.2	558	0	0.2	558	0	0.2	558	0	0.2	558	0	0.2
inithx.i.3 (559, 31)	559	0	0.2	559	0	0.2	559	0	0.2	559	0	0.2	559	0	0.2
le450-15a (450, 15)	400	0	0.1	400	0	0.2	400	0	0.4	409	20	661.3	400	2	476.6
le450-15b (450, 15)	405	0	0.1	413	230	0.1	413	230	0.4	407	6	571.8	407	6	540.3
le450-15c (450, 15)	306	0	0.1	310	156	0.3	308	141	0.5	298	10	2164.8	306	0	1849.1
le450-15d (450, 15)	307	0	0.1	308	234	0.3	309	214	0.6	302	5	2152.3	306	5	1957.7
le450-25a (450, 25)	443	0	0.1	443	0	0.1	443	0	0.3	443	0	69.9	443	0	50.6
le450-25b (450, 25)	443	0	0.1	450	357	0.1	450	327	0.3	443	0	62.2	443	0	30
le450-25c (450, 25)	389	0	0.1	399	222	0.3	399	210	0.8	393	8	994.4	389	5	953.6
le450-25d (450, 25)	389	0	0.1	404	224	0.3	402	216	0.8	391	4	805.5	390	1	743.1
le450-5a (450, 5)	261	0	0.1	261	0	0.1	261	0	0.2	262	4	1303.5	264	4	1238.7
le450-5b (450, 5)	257	0	0.1	257	0	0.1	257	0	0.2	258	22	1491.4	254	2	1285.9
le450-5c (450, 5)	196	0	0.1	196	0	0.1	196	0	0.2	213	4	2025.8	198	4	2141.4
le450-5d (450, 5)	195	0	0.1	195	0	0.1	195	0	0.2	216	6	1916.9	194	6	2115.8
mulsol.i.1 (138, 49)	138	0	0.1	138	0	0.1	138	0	0.1	138	0	0.1	138	0	0.1
mulsol.i.2 (173, 31)	173	0	0.1	173	0	0.1	173	0	0.1	173	0	0.1	173	0	0.1
mulsol.i.3 (174, 31)	174	0	0.1	174	0	0.1	174	0	0.1	174	0	0.1	174	0	0.1
mulsol.i.4 (175, 31)	175	0	0.1	175	0	0.1	175	0	0.1	175	0	0.1	175	0	0.1
mulsol.i.5 (176, 31)	176	0	0.1	176	0	0.1	176	0	0.1	176	0	0.1	176	0	0.1
zeroin.i.1 (126, 49)	126	0	0.1	126	0	0.1	126	0	0.1	126	0	0.1	126	0	0.1
zeroin.i.2 (157, 30)	157	0	0.1	157	0	0.1	157	0	0.1	157	0	0.1	157	0	0.1
zeroin.i.3 (157, 30)	157	0	0.1	157	0	0.1	157	0	0.1	157	0	0.1	157	0	0.1
anna (138, 11)	137	0	0.1	138	59	0.1	138	59	0.1	137	0	0.2	137	0	0.1
david (87, 11)	86	0	0.1	87	70	0.1	87	44	0.1	87	2	5.7	86	0	0.1
homer (556, 13)	554	0	0.1	556	207	0.1	556	207	0.2	556	6	35	555	1	17.8
huck (74, 11)	74	0	0.1	74	0	0.1	74	0	0.1	74	0	0.1	74	0	0.1
jean (77, 10)	77	0	0.1	77	0	0.1	77	0	0.1	77	0	0.1	77	0	0.1
games120 (120, 9)	120	0	0.1	120	0	0.1	120	0	0.1	120	0.1	0	120	0	0.1
miles1000 (128, 42)	126	0	0.1	126	0	0.1	126	0	0.1	126	0.1	5.9	126	0	2.8
miles1500 (128, 73)	127	0	0.1	128	106	0.1	128	53	0.3	127	0.1	7	127	0	3.2
miles250 (125, 8)	122	0	0.1	123	55	0.1	123	55	0.1	125	4	18.4	124	2	5.8
miles500 (128, 20)	127	0	0.1	128	100	0.1	128	84	0.1	128	1	12.7	127	0	0.3
miles750 (128, 31)	126	0	0.1	126	0	0.1	126	0	0.1	127	1	17.9	126	0	1.5
queen11-11 (121, 11)	91	0	0.1	94	5	0.1	94	5	0.1	95	2	57.2	100	6	29.5
queen13-13 (169, 13)	134	0	0.1	138	3	0.1	138	1	0.2	141	5	175.6	141	3	86
queen5-5 (25, 5)	20	0	0.1	20	0	0	20	0	0.1	24	7	0.1	24	4	0.2
queen6-6 (36, 7)	31	0	0.1	32	1	0	32	1	0.1	33	13	2.4	32	2	0.6
queen7-7 (49, 7)	40	0	0.1	41	1	0	41	1	0.1	42	5	2	42	1	0.6
queen8-12 (96, 12)	88	0	0.1	87	2	0	87	2	0.1	92	3	41.4	91	5	22
queen8-8 (64, 9)	55	0	0.1	55	0	0	55	0	0.1	58	11	13.2	54	2	4
queen9-9 (81, 10)	66	0	0.1	67	1	0	67	1	0.1	72	10	34.5	73	3	9.2
myciel3 (11, 4)	11	0	0.1	11	0	0.1	11	0	0.1	11	0	0.1	11	0	0.1
myciel4 (23, 5)	23	0	0.1	23	0	0.1	23	0	0.1	23	0	0.1	23	0	0.1
myciel5 (47, 6)	47	0	0.1	47	0	0.1	47	0	0.1	47	0	0.1	47	0	0.1
myciel6 (95, 7)	95	0	0.1	95	0	0.1	95	0	0.1	95	0	0.1	95	0	0.1
myciel7 (191, 8)	191	0	0.1	191	0	0.1	191	0	0.1	191	0	0.1	191	0	0.1
TOTAL	10466	0	2.3	10527	2033	3.6	10524	1851	7.1	10550	159	14650.1	10504	64	13567.6

Ces résultats illustrent bien la nécessité de considérer le TAPD et le ARPD. En effet, l'algorithme de réparation produit un meilleur TAPD (12.3%) que l'algorithme de recherche locale au premier critère d'arrêt (12.5%), mais un moins bon ARPD (8.7% pour l'un, 8.3% pour l'autre). Si l'on prend par exemple les instances `le450_25d` ($n = 450$) et `queen11_11` ($n = 121$), le premier algorithme colore respectivement 404 et 94 sommets, et le second 390 et 100. Ceci donne des TAPD et ARPD égaux à 12.8% et 16.3% pour le premier, et 14.2% et 15.2% pour le second.

Il est également pertinent d'analyser les temps d'exécution. Dans le Tableau 5.9, on observe que certaines instances sont longues à résoudre, jusqu'à 25 minutes pour la recherche locale au second critère d'arrêt, alors que l'algorithme de réparation est très rapide (moins de 0.5 secondes par instance). Ceci dit, ce n'est pas une règle générale : la recherche locale est parfois très efficace et très rapide : si l'on regarde l'instance `miles250`, on peut voir que l'algorithme de réparation colore tous les sommets sauf 2 au prix de 55 changements de couleurs, alors que la Recherche Tabou colore tous les sommets sauf 1, avec deux changements de couleur, et un temps de résolution total inférieur à 6 secondes (soit 0.6 secondes en moyenne par itération). En résumé, il y a un compromis entre les temps d'exécution et la précision pour certaines instances, et pour d'autres la recherche locale est la stratégie gagnante.

5.4.4 Régime permanent

Jusqu'à présent, nous n'avons pas considéré la possibilité qu'un sommet puisse disparaître au cours du temps. Or, dans un réseau de télécommunications, les usagers quittent le réseau et libèrent leur canal une fois leur communication terminée. Tel que mentionné dans le Chapitre 4, les fournisseurs de services mobiles doivent en principe pouvoir assurer une couverture supérieure ou égale à 98%. Nous proposons dans cette section de répondre aux questions suivantes. Lorsque le réseau est proche de ce seuil de saturation et que le trafic (le nombre d'apparitions et de disparitions d'usagers dans le réseau) augmente, comment évolue le nombre de changements de canaux ? Est-il possible de maintenir une couverture à 98% sans que ces changements augmentent drastiquement ? L'autorisation des changements de couleurs est-elle déterminante pour assurer une telle couverture ?

Pour répondre à ces questions, nous avons réalisé des tests numériques supplémentaires en se plaçant en régime permanent, ce qui signifie qu'à chaque étape de l'horizon de temps, un ensemble de sommets G_t apparaît dans le réseau, et un autre ensemble de même cardinalité F_{t-1} disparaît, de sorte qu'il y ait en permanence 50 usagers dans le réseau. Avec les mêmes notations que précédemment, $H_t = (H_{t-1} \cup G_t) \setminus F_{t-1}$ et $|H_t| = 50$ pour tout $t \in T$. Pour que $|H_t|$ soit bel et bien égal à 50, nous avons considéré que les sommets qui ne sont pas colorés ne

quittent pas le réseau et ne font donc pas partie de l'ensemble F_{t-1}. Il est possible qu'avec le départ d'autres sommets, ceux-ci puissent être colorés tardivement. Nous avons fait varier les taux d'arrivée et de départ en prenant $|F_t| = |G_t| \in \{1, 2, \cdots, 15\}$. Enfin, l'horizon de temps a été fixé à 15 périodes, et le nombre de canaux à $k = 9$, nous avons observé empiriquement que cette valeur de k permet typiquement d'assurer une couverture supérieure à 98%.

À nouveau, quatre stratégies sont comparées :
- uniquement k-LF (les changements de couleurs ne sont pas autorisés) ;
- k-LF et l'algorithme de réparation k-LF sans la permutation σ (k-LF & k-LF) ;
- k-LF et l'algorithme de réparation k-LF avec la permutation σ (k-LF & k-LF & σ) ;
- k-LF et la Recherche Tabou avec comme critère d'arrêt 10 itérations sans amélioration (k-LF & Tabou) ;
- k-LF et la Recherche Tabou avec comme critère d'arrêt 50 itérations sans amélioration (k-LF & Long Tabou).

Les résultats sont résumés dans le Tableau 5.10. Tout d'abord on constate que si l'on n'autorise aucun changement de couleur (stratégie k-LF), on peut desservir en moyenne 48 usagers, alors que les trois autres stratégies permettent de connecter un usager en plus dans le réseau. Le coût (en changements de couleurs) pour desservir cet usager supplémentaire est variable suivant l'algorithme utilisé. Plus précisément, l'algorithme de réparation produit en moyenne 34 changements de couleurs. La permutation σ permet de diminuer les changements de deux unités en moyenne. Quant à elle, la Recherche Tabou parvient à ne changer la couleur que pour 4 usagers en moyenne.

La Figure 5.13 illustre l'évolution des changements de couleur en fonction du trafic ($|G_t| = |F_t|$), exprimée en pourcentage. Par exemple, si $|G_t| = |F_t| = 15$, alors uniquement 35 usagers sur 50 sont susceptibles de changer de couleur à chaque itération. Sur un total de 15 itérations, 48 changements de couleurs donne un pourcentage égal à $100\frac{48}{35 \cdot 15} = 9.5\%$. D'après le graphique, pour l'algorithme de réparation (k-LF & k-LF), plus $|G_t| = |F_t|$ est élevé, plus le nombre de changements est élevé : lorsque $|G_t| = |F_t|$ augmente de 1 à 10, les changements augmentent de 0.15% à 7%. La permutation σ permet de diminuer très légèrement l'augmentation des changements : dans le meilleur des cas, cela permet uniquement 1% de changements en moins. En revanche, la Recherche Tabou parvient à conserver un taux de changements de l'ordre de 1%, et ceci quelque soit le trafic. De plus, les temps de calcul de la Recherche Tabou sont de l'ordre de 30 secondes, soit 2 secondes par itération en moyenne sur les 15 périodes de temps . Ceci montre que l'algorithme de réparation est plus sensible au trafic que la recherche tabou : plus le trafic est élevé, plus il est difficile de maintenir la couverture requise.

Tableau 5.10 Régime permanent, $n = 50$ usagers

	k-LF			k-LF & k-LF			k-LF & k-LF & σ			k-LF & Tabou		
$\lvert G_t\rvert = \lvert F_t\rvert$	col.	cha.	Δt	col.	cha.	Δt	col.	cha.	Δt	col.	cha.	Δt
1	46	0	0.1	46	1	0.1	46	1	0.2	49	1	20.3
2	47	0	0.1	47	6	0.1	47	6	0.2	49	2	24.1
3	47	0	0.1	48	16	0.1	48	15	0.2	49	3	27.3
4	47	0	0.1	48	16	0.1	48	16	0.2	50	3	28.1
5	47	0	0.1	48	23	0.1	48	21	0.2	49	4	35.2
6	48	0	0.1	49	25	0.1	49	24	0.1	49	5	30.9
7	48	0	0.1	49	37	0.0	49	46	0.1	50	5	33.8
8	48	0	0.1	49	50	0.1	49	43	0.1	50	4	30.6
9	48	0	0.1	49	43	0.1	49	37	0.1	50	5	34.6
10	48	0	0.1	50	44	0.1	50	44	0.1	50	4	26.2
11	49	0	0.1	50	61	0.1	50	59	0.1	50	4	26.7
12	48	0	0.1	50	44	0.1	50	37	0.1	50	3	22.9
13	48	0	0.1	50	55	0.1	49	50	0.1	50	3	25.7
14	48	0	0.1	50	41	0.1	50	40	0.1	50	3	21.4
15	49	0	0.1	50	48	0.1	50	45	0.1	50	3	22.6
Moyenne	48	0	0.1	49	34	0.1	49	32	0.1	49	4	27.4

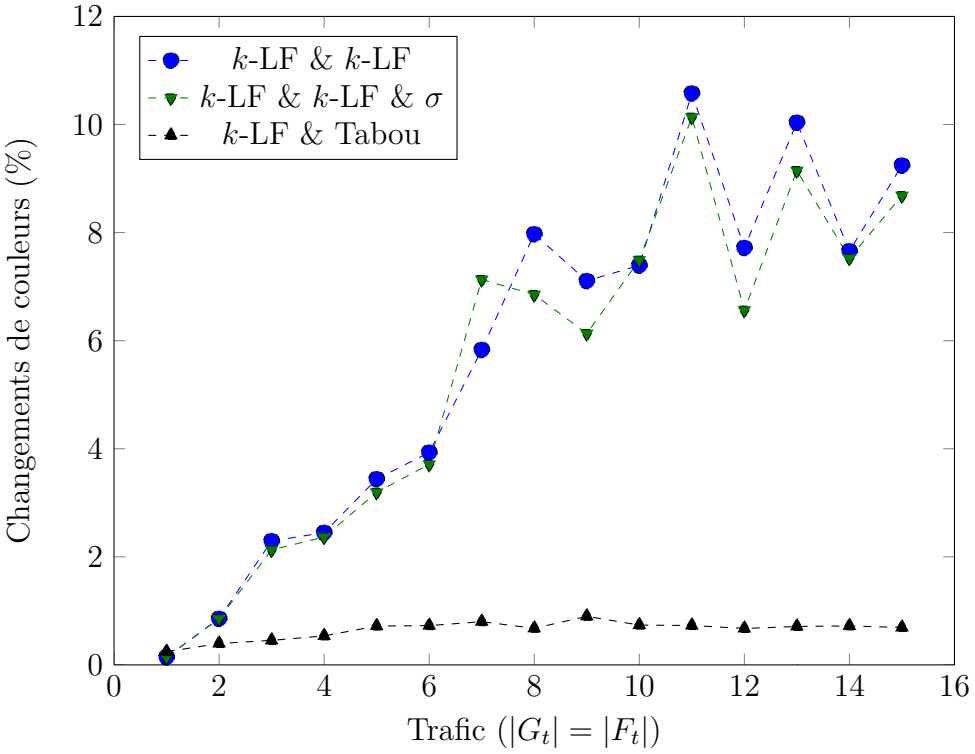


Figure 5.13 Évolution des changements de couleur en régime permanent

5.5 Retour au cas orienté, complet, pondéré, θ-impropre

5.5.1 Garanties

Les résultats de la Section 5.1 sur les graphes classiques sont généralisables sur des graphes complets, orientés, pondérés, et ceci quelque soit le paramètre θ. Il suffit de remarquer que l'on peut toujours imposer que deux sommets u et v ne puissent pas être colorés avec la même couleur en créant un arc (u, v), pondéré par n'importe quelle valeur $\omega(u, v) > \theta W(v)$. Inversement, on peut toujours s'assurer qu'ils le puissent, en pondérant tous les arcs entrant vers v et u par n'importe quelle valeur inférieure à $\frac{\theta W(v)}{n-1}$, $\frac{\theta W(u)}{n-1}$, respectivement. Autrement dit, on peut toujours construire des cliques ou des ensembles indépendants, et donc conserver la structure des graphes générés par les algorithmes *spoilers* 5.1 et 5.2.

5.5.2 Politiques θ-Fit

Il n'est pas difficile non plus d'adapter les politiques First-Fit, Next-Fit et Best-Fit au (k, W, ω, θ)-PGSI dans une configuration on-line. Il suffit de reprendre les éléments introduits au Chapitre 4 qui définissent les couleurs disponibles pour un sommet v non coloré ainsi que son degré de saturation. De plus, le caractère θ-impropre des colorations laisse place à de nouvelles politiques de priorité, voyons quelles stratégies sont envisageables.

Politique θ-Fit1

On rappelle que sur un graphe complet, orienté, pondéré sur les arcs par une fonction ω et sur les sommets par une fonction W, le degré d'un sommet v par rapport à un ensemble X donné est défini par

$$\deg_X^-(v, W, \omega) = \sum_{u \in X} \frac{\omega(u, v)}{W(v)}.$$

Cette grandeur peut être utilisée pour mesurer le niveau de saturation d'une classe de couleur et motive la définition suivante.

Définition 30. *Le niveau de saturation d'une classe V_i est défini par*

$$\xi(V_i) = \max_{v \in V_i} \left\{ \deg_{V_i \setminus \{v\}}^-(W, \omega, v) \right\}.$$

Si $\xi(V_i) = \theta$, la classe est saturée puisque tout nouveau sommet incorporé dans la classe V_i engendrerait nécessairement un conflit. Inversement, lorsque $|V_i| \leq 1$, $\xi(V_i) = 0$.

La politique θ-Fit1 pour un sommet v consiste alors à choisir la couleur i qui minimise $\xi(V_i)$, et qui est disponible. Ceci permet d'éviter d'introduire le sommet v dans une classe déjà saturée : on minimise les chances qu'un sommet déjà bien contraint soit soumis à d'avantage d'interférences.

Cette politique ne garantit pas que l'incorporation d'un sommet v dans une classe V_i peu saturée ne va pas créer de fortes interférences pour v, ou pour un sommet de V_i. Par exemple, supposons qu'une classe V_i ne contienne qu'un seul sommet u, et que le ratio $\frac{\omega(v,u)}{W(u)}$ soit proche de θ. Alors $\xi(V_i) = 0$, mais $\xi(V_i \cup v) \approx \theta$. Pour éviter ces cas de figure, nous avons développé dans le Chapitre 4 la notion de couleur *super-disponible*. Dans le même esprit, nous proposons ici une deuxième politique de choix de couleur.

Politique θ-Fit2

Rappelons qu'une couleur i est *disponible* pour un sommet non coloré v si les contraintes suivantes sont satisfaites :

$$\begin{aligned} \sum_{u \in V_i} \omega(u,v) &\leq \theta\, W(v) \\ \omega(v,u) + \sum_{x \in V_i | x \neq u} \omega(x,u) &\leq \theta\, W(u) \qquad \forall u \in V_i. \end{aligned}$$

En s'appuyant sur cette définition, il est possible de quantifier le niveau de disponibilité de cette couleur.

Définition 31. *Le niveau de disponibilité d'une couleur i pour un sommet v non coloré est défini par*

$$\theta_i(v) = \max\left\{ \sum_{u \in V_i} \frac{\omega(u,v)}{W(v)}, \max_{u \in V_i} \left\{ \frac{\omega(v,u)}{W(u)} + \sum_{x \in V_i | x \neq u} \frac{\omega(x,u)}{W(u)} \right\} \right\}$$

Le premier terme entre crochets représente le niveau d'interférences que subirait le sommet v s'il intégrait la classe V_i, et le second terme représente l'interférence maximale que subirait un sommet déjà présent dans V_i. Ainsi, la couleur i est disponible pour le sommet v si et seulement si $\theta_i(v) \leq \theta$. Au Chapitre 4, les couleurs prioritaires obtenues à l'aide de diagrammes de Voronoï et les couleurs super-disponibles étaient deux façons de déterminer des couleurs avec un coefficient $\theta_i(v)$ assez faible.

La politique θ-Fit2 consiste à choisir la couleur i la plus disponible, i.e. celle qui minimise $\theta_i(v)$.

Politique θ-Fit3

Les politiques θ-Fit1 et θ-Fit2 donnent la priorité aux classes de couleurs peu saturées, mais ne tiennent pas compte des autres sommets non colorés de l'ensemble G_t. Il peut être plus judicieux de réserver ces classes pour des sommets difficiles à colorer. La politique Best-Fit est particulièrement intéressante car justement, elle prend en compte les degrés de saturation des sommets non colorés et donne la priorité à la couleur qui diminue le degré de saturation pour un minimum d'entre eux. Rappelons que le degré de saturation d'un sommet v est défini comme le nombre de couleurs non disponibles pour v :

$$\mathrm{dsat}(v, W, \omega) = |\{i \in \{1, \cdots, k\} \mid \theta_i(v) > \theta\}|.$$

Pour affiner ce choix, nous proposons de départager les cas d'égalité en calculant l'impact du choix d'une couleur sur les niveaux de disponibilité de cette couleur pour les autres sommets de G_t pour lesquels celle-ci demeure disponible. Étant donnée une couleur i disponible pour deux sommets v et y, on peut quantifier l'impact d'assigner la couleur i à v sur $\theta_i(y)$ en calculant

$$\tilde{\theta}_i(v, y) = \max\left\{\frac{\omega(v,y)}{W(y)} + \sum_{u \in V_i} \frac{\omega(u,y)}{W(y)}, \max_{u \in V_i}\left\{\frac{\omega(v,u)}{W(u)} + \frac{\omega(y,u)}{W(u)} + \sum_{x \in V_i | x \neq u} \frac{\omega(x,u)}{W(u)}\right\}\right\}.$$

On reconnaît les mêmes termes que pour $\theta_i(y)$, auquels sont ajoutés les contributions du sommet v, à savoir $\frac{\omega(v,y)}{W(y)}$ et $\frac{\omega(v,u)}{W(u)}$. Si cette grandeur est strictement plus grande que θ, la couleur n'est plus disponible pour y dès lors que v prend la couleur i, et son degré de saturation augmente d'une unité. La politique θ-Fit3 pour un sommet v départage les cas d'égalité de Best-Fit en sélectionnant la couleur qui minimise l'impact moyen sur les autres sommets non colorés pour lesquels elle demeure disponible, i.e. celle qui minimise

$$\sum_{y \in G_t \mid \theta_i(y) \leq \theta} \tilde{\theta}_i(v, y).$$

5.5.3 Résultats numériques

Comparaison des politiques avec la solution optimale

Dans un premier temps, nous avons comparé ces 6 politiques (First-Fit, Next-Fit, Best-Fit, θ-Fit1, θ-Fit2, θ-Fit3) couplées avec l'algorithme WP1 du Chapitre 4 avec la solution optimale sur une série de 100 instances générées de façon aléatoire, de la même manière qu'au Chapitre 4, mais en faisant apparaître des paquets de sommets itérativement. La solution optimale est calculée par programmation en nombres entiers. Sur ces 100 instances, le nombre d'usagers a été fixé à $n = 25$, le nombre de stations de base à $t = 5$, le nombre de canaux à $k = 9$, la taille des paquets des sommets révélés à $|G_t| = 5$ et θ à 0.25. Les résultats sont résumés à la Figure 5.14 sous forme de profils, avec en abscisse l'erreur relative maximale admise et en ordonnée le pourcentage d'instances résolues en deça de cette erreur relative.

Tout d'abord, on constate sans surprise que comme sur les graphes non orientés, non pondérés, Best-Fit est la meilleure stratégie, avec 30% d'instances résolues de façon optimale. First-Fit fait presque aussi bien, avec 28% d'instances résolues à l'optimalité, et une courbe légèrement en deça de celle de Best-Fit de manière générale. La stratégie Next-Fit résout 10% d'instances de façon optimale et environ 10% d'instances de moins que Best-Fit, quelque soit l'erreur relative maximale tolérée. Concernant les politiques θ-Fit, on observe que θ-Fit1 et θ-Fit2 se comportent de manière similaire à Next-Fit, et θ-Fit3 est identique à Best-Fit.

Ces observations ne sont pas étonnantes, car les stratégies θ-Fit1 et θ-Fit2 donnent la priorité aux classes de couleur les moins saturées et les plus disponibles, respectivement. Ainsi, les k classes de couleurs sont non vides après l'apparition des k premiers sommets, privant ainsi les sommets éventuellement difficiles à colorer d'une couleur disponible. Inversement, First-Fit sature séquentiellement les classes de couleur, de façon à laisser plus de couleurs disponibles pour les sommets suivants. Clairement cette stratégie est plus efficace pour colorer un maximum de couleurs. Les autres seraient plus adapatées pour maintenir un niveau d'interférences global minimal.

Pour affiner la comparaison entre Best-Fit et θ-Fit3, nous avons effectué une autre série de simulations avec les mêmes paramètres, mais avec $|G_t| = 25$, de façon à avoir un scénario off-line. Les résultats sont représentés à la Figure 5.15. On observe que les courbes sont pratiquement superposées, celle de θ-Fit3 est très légèrement au-dessus lorsque l'erreur relative maximale tolérée est comprise entre 5 et 9%. Force est de constater que le fait de départager les cas d'égalité de Best-Fit en donnant la priorité à la couleur qui a le moins d'impact sur les niveaux de disponiblités des sommets pour lesquels elle demeure disponible n'a pas beaucoup d'influence sur la solution.

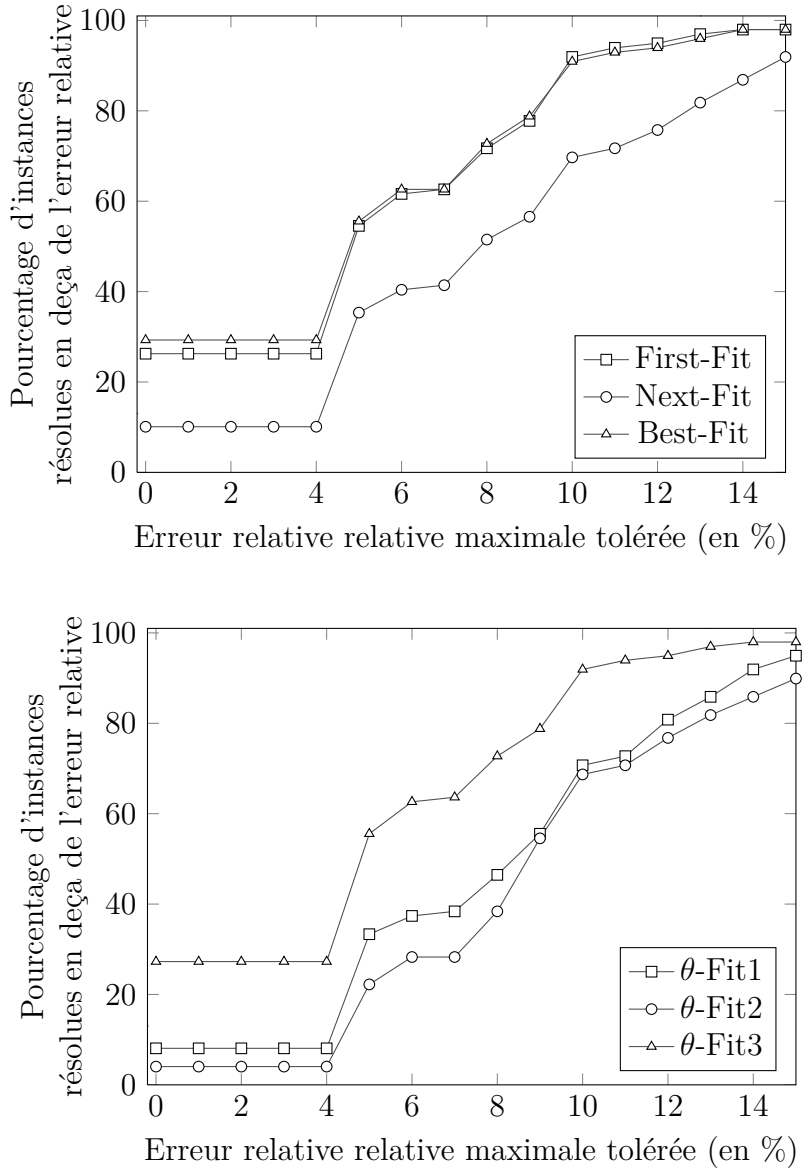


Figure 5.14 Comparaison des 6 politiques avec la solution optimale, $(n, |G_t|, k, \theta) = (25, 5, 9, 0.25)$

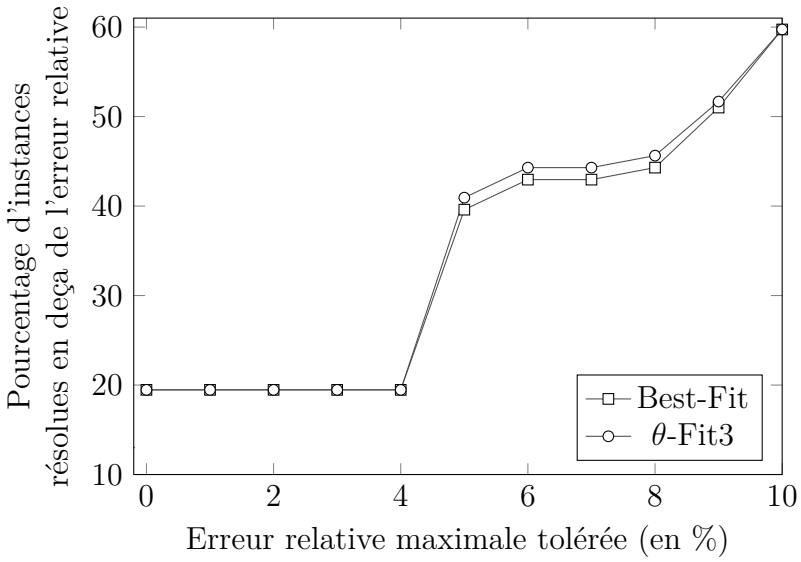


Figure 5.15 Comparaison de Best-Fit et θ-Fit3 avec la solution optimale, $(n, |G_t|, k, \theta) = (25, 25, 9, 0.25)$

5.6 Conclusion

Dans ce chapitre, nous avons étendu le modèle de coloration du $(k, 1, 1, 0)$-PGSI sur un horizon de temps dans une configuration on-line : les décisions sont irrévocables et la topologie du réseau n'est pas connue d'avance.

Nous avons analysé les meilleurs garanties possibles pour ce problème et avons proposé une série d'algorithmes pour le traiter, ainsi qu'une de ses relaxations, où les sommets colorés peuvent changer de couleur moyennant un coût.

Pour la version on-line pure, les résultats expérimentaux semblent montrer que la meilleure stratégie serait celle basée sur l'algorithme SLF, avec les règles de priorité Best-Fit, ou First-Fit. En saturant séquentiellement les classes de couleurs, First-Fit laisse plus de couleurs disponibles pour les sommets suivants. Quant à la politique Best-Fit, elle considère les autres sommets non colorés et s'assure de laisser un maximum de couleurs disponibles également. Tel que l'on peut se l'imaginer, les tests numériques ont montré que plus l'horizon de temps est grand (ou plus la taille des paquets est petite), plus il est difficile de colorer un maximum de sommets.

Nous avons également testé ces stratégies sur des graphes complets, orientés, et pondérés et avons obtenu les mêmes conclusions. De plus, nous avons proposé trois autres politiques de priorité spécifiquement dédiées à la coloration θ-impropre appelées θ-Fit$i$ $(i = 1, 2, 3)$. Force

est de constater que celles-ci se sont révélées moins efficaces que Best-Fit, excepté θ-Fit3 qui l'améliore très légèrement en départageant les cas d'égalité de façon à limiter l'impact moyen sur les niveaux de disponibilités des couleurs potentielles pour les sommets en attente d'une couleur.

Pour la relaxation, nous avons comparé un algorithme de réparation et un algorithme de recherche locale. Celle-ci s'avère plus intéressante dans la mesure où elle donne de meilleurs résultats vis-à-vis des deux objectifs conflictuels considérés, mais au prix de longs calculs sur certaines instances particulières. La recherche locale possède de nombreux avantages en pratique. Sa flexibilité fait d'elle une stratégie pouvant être adaptée à différentes circonstances. Par exemple, il est possible de contrôler les temps de calculs en jouant sur le critère d'arrêt, donnant ainsi la priorité à l'un ou l'autre des deux objectifs : de longs temps de calculs augmenteront le nombre de sommets colorés, mais aussi le nombre de changements de couleurs, tandis qu'un parcours rapide de l'espace de recherche aura l'effet inverse. De plus, la recherche locale s'est avérée plus intéressante en régime permanent : elle permet de colorer 98% de sommets quelque soit le trafic dans le réseau, en maintenant des taux de changements de couleurs faibles.

Passons maintenant à l'ultime chapitre de cette thèse, afin de récapituler l'ensemble des contributions proposées ici, ainsi que les limites de ces travaux et les axes de recherche qui en découlent.

Tableau 5.11 Les sommets arrivent un par un

Couleurs k / INSTANCE (n, χ)	χ col.	Δt	$\chi+1$ col.	Δt	$\chi+2$ col.	Δt	$\chi+3$ col.	Δt	$\chi+4$ col.	Δt	$\chi+5$ col.	Δt	$\chi+6$ col.	Δt	$\chi+7$ col.	Δt	$\chi+8$ col.	Δt	$\chi+9$ col.	Δt	$\chi+10$ col.	Δt
`fpsol2.i.1` (269, 65)																						
First-Fit	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5	**269**	1.5
Next-Fit	261	1.4	262	1.4	263	1.4	264	1.4	265	1.4	266	1.5	267	1.5	268	1.5	**269**	1.5	**269**	1.5	**269**	1.5
`fpsol2.i.2` (363, 30)																						
First-Fit	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0
Next-Fit	360	1.9	360	1.9	360	1.9	361	1.9	361	1.9	361	1.9	361	1.9	361	1.9	362	1.9	362	1.9	362	1.9
`fpsol2.i.3` (363, 30)																						
First-Fit	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0	**363**	2.0
Next-Fit	360	1.9	360	1.9	360	1.9	361	1.9	361	1.9	361	1.9	361	1.9	361	1.9	362	1.9	362	1.9	362	1.9
`inithx.i.1` (519, 54)																						
First-Fit	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.8	**519**	5.7	**519**	5.8	**519**	5.8
Next-Fit	509	5.5	509	5.5	509	5.5	510	5.5	510	5.5	510	5.5	510	5.5	510	5.5	510	5.5	510	5.5	510	5.5
`inithx.i.2` (558, 31)																						
First-Fit	**558**	5.3	**558**	5.2	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3	**558**	5.3
Next-Fit	553	5.2	554	5.2	555	5.2	555	5.2	555	5.2	555	5.2	555	5.2	556	5.2	556	5.2	556	5.2	**558**	5.2
`inithx.i.3` (559, 31)																						
First-Fit	**559**	5.6	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3	**559**	5.3
Next-Fit	554	5.2	555	5.2	556	5.3	556	5.2	556	5.2	556	5.2	556	5.2	557	5.2	557	5.2	557	5.2	**559**	5.3
`le450-15a` (450, 15)																						
First-Fit	406	2.4	418	2.4	429	2.4	437	2.4	442	2.4	447	2.4	449	2.4	**450**	2.4	**450**	2.5	**450**	2.4	**450**	2.4
Next-Fit	383	2.5	398	2.3	413	2.6	405	2.3	426	3.1	423	2.4	433	2.5	430	2.4	441	2.4	442	2.5	442	2.4
`le450-15b` (450, 15)																						
First-Fit	404	2.4	416	2.5	427	2.3	434	2.4	442	2.4	445	2.4	448	2.4	**450**	2.4	**450**	2.4	**450**	2.4	**450**	2.4
Next-Fit	391	2.3	400	2.3	404	2.3	414	2.3	426	2.3	425	2.4	432	2.4	432	2.4	437	2.4	443	2.4	439	2.3
`le450-15c` (450, 15)																						
First-Fit	298	3.4	311	3.4	327	3.4	340	3.5	351	3.7	366	3.5	378	3.5	388	3.7	398	3.6	408	3.7	417	3.6
Next-Fit	297	3.3	311	3.3	326	3.4	328	3.8	342	3.7	357	3.4	367	3.6	374	3.5	381	3.5	392	3.6	396	4.1
`le450-15d` (450, 15)																						
First-Fit	305	3.3	319	3.4	333	3.4	348	3.5	362	3.6	374	3.6	383	3.6	396	3.7	404	3.8	413	3.7	421	3.7
Next-Fit	300	3.6	309	3.3	320	3.4	335	3.7	343	3.6	351	3.5	368	3.6	378	3.8	381	3.6	386	3.6	393	3.6
`le450-25a` (450, 25)																						
First-Fit	441	3.0	444	2.4	446	2.5	449	2.4	**450**	2.4	**450**	2.4	**450**	2.5	**450**	2.5	**450**	2.5	**450**	2.4	**450**	2.5
Next-Fit	426	2.4	433	2.4	435	2.5	432	2.4	439	2.4	444	2.5	440	2.5	445	2.4	442	2.4	445	2.6	445	2.4
`le450-25b` (450, 25)																						
First-Fit	442	2.4	445	2.4	448	2.4	**450**	2.6	**450**	2.4	**450**	2.4	**450**	2.4	**450**	2.4	**450**	2.5	**450**	2.4	**450**	2.4
Next-Fit	431	2.4	432	2.4	434	2.4	436	2.4	439	2.4	441	2.4	442	2.5	445	2.4	448	2.4	445	2.5	446	2.3
`le450-25c` (450, 25)																						
First-Fit	388	3.8	396	4.1	403	3.8	411	4.1	419	3.8	424	3.9	431	4.0	436	3.9	440	3.9	443	4.0	448	3.9
Next-Fit	375	3.8	383	4.3	388	3.9	397	3.8	396	3.8	403	3.8	408	3.9	417	3.8	418	3.9	426	3.8	426	3.8
`le450-25d` (450, 25)																						
First-Fit	394	4.0	401	3.8	409	3.8	418	8.0	426	3.9	432	3.9	438	4.0	441	4.0	446	3.9	448	3.9	449	3.9
Next-Fit	379	3.8	380	3.9	390	7.9	394	4.2	398	3.8	404	3.9	407	3.9	415	4.1	416	3.8	426	3.8	424	4.1
`le450-5a` (450, 5)																						
First-Fit	261	1.8	305	1.8	343	1.9	375	1.9	406	2.1	427	2.0	439	2.0	447	2.0	**450**	2.0	**450**	2.1	**450**	2.0
Next-Fit	260	1.8	289	1.8	320	1.9	351	1.9	375	1.9	394	2.0	418	2.0	434	2.0	434	2.1	441	2.0	446	2.1
`le450-5b` (450, 5)																						
First-Fit	260	1.8	303	1.9	342	1.9	380	1.9	407	2.0	426	2.0	446	2.0	**450**	2.0	**450**	2.0	**450**	2.0	**450**	2.0
Next-Fit	249	1.8	281	1.9	324	1.9	351	3.1	389	1.9	401	2.0	410	2.0	425	2.0	443	2.0	443	2.0	447	2.2
`le450-5c` (450, 5)																						
First-Fit	215	2.2	247	2.3	283	2.3	311	2.4	333	2.4	358	2.5	383	2.5	408	2.6	426	2.6	440	2.6	448	2.7
Next-Fit	208	2.1	229	2.2	246	2.3	304	2.4	318	2.4	334	2.5	349	3.1	371	2.5	378	2.6	401	2.9	409	2.6
`le450-5d` (450, 5)																						
First-Fit	206	2.2	240	2.8	272	2.3	301	2.4	329	2.6	354	2.5	382	2.6	400	2.6	423	2.7	440	2.9	447	2.6
Next-Fit	193	2.1	231	2.2	266	2.3	299	2.7	317	2.4	337	2.4	381	2.5	371	2.8	391	2.5	407	2.7	415	2.6
`mulsol.i.1` (138, 49)																						
First-Fit	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3
Next-Fit	136	0.3	137	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3	**138**	0.3
`mulsol.i.2` (173, 31)																						
First-Fit	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4
Next-Fit	170	0.4	172	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4	**173**	0.4
`mulsol.i.3` (174, 31)																						
First-Fit	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4
Next-Fit	171	0.4	173	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4	**174**	0.4
`mulsol.i.4` (175, 31)																						
First-Fit	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4
Next-Fit	172	0.4	174	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4	**175**	0.4
`mulsol.i.5` (176, 31)																						
First-Fit	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4
Next-Fit	174	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4	**176**	0.4
`zeroin.i.1` (126, 49)																						
First-Fit	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2
Next-Fit	123	0.2	124	0.2	125	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2
`zeroin.i.2` (157, 30)																						
First-Fit	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3
Next-Fit	146	0.3	155	0.3	155	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3

Tableau 5.11 Les sommets arrivent un par un (suite et fin)

Couleurs k	χ		$\chi+1$		$\chi+2$		$\chi+3$		$\chi+4$		$\chi+5$		$\chi+6$		$\chi+7$		$\chi+8$		$\chi+9$		$\chi+10$	
INSTANCE (n,χ)	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt
`zeroin.i.3` $(157,30)$																						
First-Fit	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3
Next-Fit	146	0.3	155	0.3	155	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3	**157**	0.3
`anna` $(138,11)$																						
First-Fit	137	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1
Next-Fit	135	0.1	132	0.1	135	0.1	**138**	0.1	137	0.1	136	0.1	**138**	0.1	137	0.1	137	0.1	**138**	0.1	**138**	0.1
`david` $(87,11)$																						
First-Fit	86	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
Next-Fit	84	0.0	85	0.0	86	0.0	85	0.0	85	0.0	86	0.0	86	0.0	86	0.1	86	0.0	86	0.0	86	0.0
`homer` $(556,13)$																						
First-Fit	554	2.0	555	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0	**556**	2.0
Next-Fit	546	2.0	546	2.0	551	2.0	553	2.0	551	2.0	553	2.0	554	2.0	553	2.0	555	2.0	555	2.0	555	2.0
`huck` $(74,11)$																						
First-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
Next-Fit	73	0.0	73	0.0	73	0.0	73	0.0	73	0.0	73	0.0	73	0.0	73	0.0	**74**	0.0	**74**	0.0	73	0.0
`jean` $(77,10)$																						
First-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
Next-Fit	76	0.0	76	0.0	76	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
`games120` $(120,9)$																						
First-Fit	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1
Next-Fit	117	0.1	119	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1	**120**	0.1
`miles1000` $(128,42)$																						
First-Fit	126	0.2	127	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2	**128**	0.2
Next-Fit	119	0.2	120	0.2	123	0.2	123	0.2	124	0.2	124	0.2	123	0.2	124	0.2	126	0.2	124	0.2	127	0.2
`miles1500` $(128,73)$																						
First-Fit	125	0.3	126	0.3	127	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3
Next-Fit	119	0.3	119	0.3	121	0.3	121	0.3	122	0.3	121	0.3	124	0.3	125	0.2	126	0.3	126	0.3	127	0.3
`miles250` $(125,8)$																						
First-Fit	124	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1
Next-Fit	123	0.1	123	0.1	124	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1	**125**	0.1
`miles500` $(128,20)$																						
First-Fit	126	0.1	127	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
Next-Fit	124	0.1	125	0.1	124	0.1	125	0.1	126	0.1	127	0.1	127	0.1	126	0.1	126	0.1	**128**	0.1	**128**	0.1
`miles750` $(128,31)$																						
First-Fit	125	0.1	126	0.1	127	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
Next-Fit	123	0.2	123	0.1	124	0.1	124	0.1	125	0.1	125	0.1	125	0.1	125	0.1	127	0.1	126	0.1	127	0.1
`queen11-11` $(121,11)$																						
First-Fit	93	0.1	100	0.1	106	0.1	111	0.1	114	0.1	118	0.1	**121**	0.1	**121**	0.1	**121**	0.1	**121**	0.1	**121**	0.1
Next-Fit	**121**	0.1	**121**	0.1	**121**	0.1	**121**	0.1	120	0.1	119	0.1	**121**	0.1	**121**	0.1	**121**	0.1	**121**	0.1	**121**	0.1
`queen13-13` $(169,13)$																						
First-Fit	130	0.3	138	0.3	144	0.3	152	0.3	158	0.3	163	0.3	165	0.3	167	0.3	**169**	0.3	**169**	0.3	**169**	0.3
Next-Fit	**169**	0.3	164	0.3	154	0.3	162	0.3	**169**	0.3	167	0.3	**169**	0.3	**169**	0.3	**169**	0.3	**169**	0.3	**169**	0.3
`queen5-5` $(25,5)$																						
First-Fit	20	0.0	23	0.0	24	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
Next-Fit	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
`queen6-6` $(36,7)$																						
First-Fit	30	0.0	33	0.0	34	0.0	35	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
Next-Fit	30	0.0	35	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
`queen7-7` $(49,7)$																						
First-Fit	39	0.0	43	0.0	47	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
Next-Fit	**49**	0.0	48	0.0	45	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
`queen8-12` $(96,12)$																						
First-Fit	86	0.1	90	0.1	94	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1
Next-Fit	91	0.1	93	0.1	93	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1	**96**	0.1
`queen8-8` $(64,9)$																						
First-Fit	56	0.0	60	0.0	62	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
Next-Fit	57	0.0	62	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
`queen9-9` $(81,10)$																						
First-Fit	68	0.1	72	0.0	75	0.1	77	0.1	79	0.1	80	0.1	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.1	**81**	0.1
Next-Fit	71	0.0	**81**	0.0	79	0.0	**81**	0.1	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.1	**81**	0.1	**81**	0.1	**81**	0.0
`myciel3` $(11,4)$																						
First-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
Next-Fit	**11**	0.0	10	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
`myciel4` $(23,5)$																						
First-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
Next-Fit	20	0.0	22	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	22	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
`myciel5` $(47,6)$																						
First-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
Next-Fit	44	0.0	46	0.0	**47**	0.0	46	0.0	**47**	0.0	44	0.0	46	0.0	**47**	0.0	46	0.0	**47**	0.0	**47**	0.0
`myciel6` $(95,7)$																						
First-Fit	**95**	0.1	**95**	0.1	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.1	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.1	**95**	0.0
Next-Fit	92	0.1	94	0.0	92	0.1	94	0.1	88	0.1	92	0.1	94	0.1	92	0.1	94	0.0	**95**	0.1	**95**	0.1
`myciel7` $(191,8)$																						
First-Fit	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3	**191**	0.3
Next-Fit	188	0.3	184	0.3	188	0.3	176	0.3	184	0.3	188	0.3	184	0.3	188	0.3	190	0.3	**191**	0.3	190	0.3

Tableau 5.12 Les sommets sont révélés en une itération

Couleurs k	χ		$\chi+1$		$\chi+2$		$\chi+3$		$\chi+4$		$\chi+5$		$\chi+6$		$\chi+7$		$\chi+8$		$\chi+9$		$\chi+10$	
INSTANCE (n,χ)	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt
`fpsol2.i.1` $(269,65)$																						
k-LF & Best-Fit	**269**	0.5	**269**	0.5	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6
k-LF & First-Fit	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1
k-LF & Next-Fit	266	0.0	267	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0	**269**	0.0
k-SLF & Best-Fit	**269**	0.5	**269**	0.5	**269**	0.5	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6	**269**	0.6
k-SLF & First-Fit	**269**	0.2	**269**	0.1	**269**	0.1	**269**	0.2	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1
k-SLF & Next-Fit	266	0.1	267	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1	**269**	0.1
`fpsol2.i.2` $(363,30)$																						
k-LF & Best-Fit	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.3	**363**	0.3	**363**	0.3	**363**	0.3
k-LF & First-Fit	**363**	0.0	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.0	**363**	0.1	**363**	0.0	**363**	0.0	**363**	0.1	**363**	0.0
k-LF & Next-Fit	344	0.0	358	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0
k-SLF & Best-Fit	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.3	**363**	0.3	**363**	0.3
k-SLF & First-Fit	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1
k-SLF & Next-Fit	358	0.1	358	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1
`fpsol2.i.3` $(363,30)$																						
k-LF & Best-Fit	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.3	**363**	0.3	**363**	0.3	**363**	0.3
k-LF & First-Fit	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.0	**363**	0.1	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0
k-LF & Next-Fit	344	0.0	358	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0	**363**	0.0
k-SLF & Best-Fit	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.2	**363**	0.3	**363**	0.3
k-SLF & First-Fit	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1
k-SLF & Next-Fit	358	0.1	358	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1	**363**	0.1
`inithx.i.1` $(519,54)$																						
k-LF & Best-Fit	**519**	0.7	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.9	**519**	0.9	**519**	0.9	**519**	0.9
k-LF & First-Fit	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.2	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1
k-LF & Next-Fit	507	0.1	513	0.1	516	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1
k-SLF & Best-Fit	**519**	0.7	**519**	0.7	**519**	0.7	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.8	**519**	0.9
k-SLF & First-Fit	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2	**519**	0.2
k-SLF & Next-Fit	516	0.1	516	0.1	516	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1	**519**	0.1
`inithx.i.2` $(558,31)$																						
k-LF & Best-Fit	**558**	0.3	**558**	0.3	**558**	0.3	**558**	0.3	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4
k-LF & First-Fit	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1
k-LF & Next-Fit	545	0.1	**558**	0.0	**558**	0.0	**558**	0.1	**558**	0.0	**558**	0.0	**558**	0.0	**558**	0.0	**558**	0.0	**558**	0.0	**558**	0.0
k-SLF & Best-Fit	**558**	0.3	**558**	0.3	**558**	0.3	**558**	0.3	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4	**558**	0.4
k-SLF & First-Fit	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1
k-SLF & Next-Fit	557	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1	**558**	0.1
`inithx.i.3` $(559,31)$																						
k-LF & Best-Fit	**559**	0.3	**559**	0.3	**559**	0.3	**559**	0.3	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4
k-LF & First-Fit	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1
k-LF & Next-Fit	546	0.1	**559**	0.1	**559**	0.0	**559**	0.0	**559**	0.0	**559**	0.0	**559**	0.0	**559**	0.0	**559**	0.1	**559**	0.0	**559**	0.0
k-SLF & Best-Fit	**559**	0.3	**559**	0.3	**559**	0.3	**559**	0.3	**559**	0.3	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4	**559**	0.4
k-SLF & First-Fit	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1
k-SLF & Next-Fit	558	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1	**559**	0.1
`le450-15a` $(450,15)$																						
k-LF & Best-Fit	423	0.1	442	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2
k-LF & First-Fit	423	0.0	440	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-LF & Next-Fit	410	0.0	423	0.0	434	0.0	442	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-SLF & Best-Fit	446	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2
k-SLF & First-Fit	437	0.1	445	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	438	0.1	446	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-15b` $(450,15)$																						
k-LF & Best-Fit	428	0.1	444	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2
k-LF & First-Fit	424	0.0	441	0.0	448	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.1
k-LF & Next-Fit	414	0.0	428	0.0	441	0.0	447	0.0	448	0.0	**450**	0.0	**450**	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-SLF & Best-Fit	446	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2
k-SLF & First-Fit	445	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	439	0.1	448	0.1	448	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-15c` $(450,15)$																						
k-LF & Best-Fit	296	0.2	318	0.2	337	0.2	359	0.2	373	0.2	395	0.2	408	0.2	425	0.2	437	0.3	445	0.3	448	0.3
k-LF & First-Fit	293	0.1	313	0.1	330	0.1	346	0.1	366	0.1	383	0.1	400	0.1	417	0.1	430	0.1	443	0.1	447	0.1
k-LF & Next-Fit	294	0.1	303	0.1	321	0.1	337	0.1	366	0.1	376	0.1	394	0.1	402	0.1	414	0.1	426	0.1	434	0.1
k-SLF & Best-Fit	313	0.1	325	0.2	341	0.2	355	0.2	384	0.2	394	0.2	408	0.2	434	0.2	445	0.3	**450**	0.3	**450**	0.3
k-SLF & First-Fit	300	0.1	324	0.1	350	0.1	365	0.1	393	0.1	400	0.1	419	0.1	433	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	309	0.1	331	0.1	334	0.1	357	0.1	384	0.1	404	0.1	417	0.1	439	0.1	440	0.1	**450**	0.1	**450**	0.1
`le450-15d` $(450,15)$																						
k-LF & Best-Fit	292	0.2	310	0.2	332	0.2	350	0.2	372	0.2	384	0.2	404	0.2	420	0.2	434	0.3	442	0.3	**450**	0.3
k-LF & First-Fit	290	0.1	307	0.1	323	0.1	344	0.1	362	0.1	380	0.1	396	0.1	412	0.1	425	0.1	436	0.1	445	0.1
k-LF & Next-Fit	281	0.0	320	0.1	323	0.1	341	0.1	349	0.1	379	0.1	398	0.1	399	0.1	414	0.1	424	0.1	424	0.1
k-SLF & Best-Fit	296	0.1	319	0.2	346	0.2	364	0.2	380	0.2	394	0.2	414	0.2	427	0.2	448	0.3	**450**	0.3	**450**	0.3
k-SLF & First-Fit	309	0.1	320	0.1	338	0.1	359	0.1	376	0.1	391	0.1	411	0.1	437	0.1	449	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	300	0.1	312	0.1	341	0.1	357	0.1	376	0.1	398	0.1	414	0.1	442	0.1	447	0.1	449	0.1	**450**	0.1
`le450-25a` $(450,25)$																						
k-LF & Best-Fit	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.3	**450**	0.3
k-LF & First-Fit	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.1	**450**	0.0	**450**	0.1	**450**	0.0	**450**	0.0	**450**	0.1
k-LF & Next-Fit	448	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-SLF & Best-Fit	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.3
k-SLF & First-Fit	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-25b` $(450,25)$																						
k-LF & Best-Fit	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.3
k-LF & First-Fit	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.1	**450**	0.0	**450**	0.0	**450**	0.1
k-LF & Next-Fit	449	0.0	**450**	0.0	449	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-SLF & Best-Fit	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.2	**450**	0.3
k-SLF & First-Fit	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1

Tableau 5.12 Les sommets sont révélés en une itération (suite)

Couleurs k	χ		$\chi+1$		$\chi+2$		$\chi+3$		$\chi+4$		$\chi+5$		$\chi+6$		$\chi+7$		$\chi+8$		$\chi+9$		$\chi+10$	
INSTANCE (n, χ)	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt
`le450-25c` $(450, 25)$																						
k-LF & Best-Fit	402	0.3	417	0.3	430	0.3	441	0.3	445	0.4	**450**	0.4	**450**	0.4	**450**	0.4	**450**	0.4	**450**	0.4	**450**	0.4
k-LF & First-Fit	403	0.1	417	0.1	430	0.1	443	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-LF & Next-Fit	397	0.1	407	0.1	411	0.1	428	0.1	429	0.1	439	0.1	444	0.1	446	0.1	447	0.1	449	0.1	**450**	0.1
k-SLF & Best-Fit	410	0.2	434	0.3	446	0.3	**450**	0.3	**450**	0.3	**450**	0.3	**450**	0.3	**450**	0.3	**450**	0.4	**450**	0.4	**450**	0.4
k-SLF & First-Fit	410	0.1	424	0.1	439	0.1	447	0.2	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	415	0.1	429	0.1	443	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-25d` $(450, 25)$																						
k-LF & Best-Fit	406	0.3	419	0.3	429	0.3	441	0.3	447	0.3	449	0.3	**450**	0.3	**450**	0.4	**450**	0.4	**450**	0.4	**450**	0.4
k-LF & First-Fit	395	0.1	409	0.1	424	0.1	437	0.1	446	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-LF & Next-Fit	398	0.1	406	0.1	411	0.1	426	0.1	434	0.1	438	0.1	445	0.1	445	0.1	446	0.1	449	0.1	**450**	0.1
k-SLF & Best-Fit	421	0.2	435	0.3	444	0.3	449	0.3	**450**	0.3	**450**	0.3	**450**	0.3	**450**	0.4	**450**	0.4	**450**	0.4	**450**	0.4
k-SLF & First-Fit	422	0.1	438	0.1	443	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	413	0.1	439	0.1	444	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-5a` $(450, 5)$																						
k-LF & Best-Fit	266	0.1	315	0.1	364	0.1	402	0.1	427	0.1	445	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-LF & First-Fit	255	0.0	299	0.0	341	0.0	382	0.0	416	0.0	438	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-LF & Next-Fit	243	0.0	287	0.0	331	0.0	370	0.0	396	0.0	427	0.0	441	0.0	441	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-SLF & Best-Fit	255	0.1	331	0.1	373	0.1	415	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & First-Fit	260	0.1	311	0.0	357	0.0	410	0.0	439	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	262	0.0	326	0.1	358	0.1	408	0.1	440	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.0	**450**	0.1	**450**	0.0
`le450-5b` $(450, 5)$																						
k-LF & Best-Fit	262	0.1	301	0.1	346	0.1	386	0.1	428	0.1	440	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-LF & First-Fit	248	0.0	295	0.0	336	0.0	374	0.0	407	0.0	436	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-LF & Next-Fit	244	0.0	294	0.0	352	0.0	369	0.0	395	0.0	426	0.0	432	0.0	447	0.0	449	0.0	448	0.0	**450**	0.0
k-SLF & Best-Fit	278	0.0	386	0.1	433	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & First-Fit	313	0.0	358	0.0	415	0.1	445	0.1	**450**	0.1	**450**	0.1	**450**	0.0	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	276	0.1	338	0.1	372	0.1	408	0.1	438	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-5c` $(450, 5)$																						
k-LF & Best-Fit	403	0.1	434	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2
k-LF & First-Fit	317	0.0	358	0.0	386	0.0	407	0.0	426	0.0	440	0.0	449	0.0	**450**	0.0	**450**	0.0	**450**	0.0	**450**	0.0
k-LF & Next-Fit	212	0.0	235	0.0	267	0.0	295	0.0	318	0.0	332	0.0	360	0.0	409	0.0	402	0.0	411	0.0	436	0.0
k-SLF & Best-Fit	255	0.1	317	0.1	365	0.1	395	0.1	425	0.1	446	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & First-Fit	317	0.1	365	0.1	354	0.1	431	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	255	0.0	315	0.1	394	0.1	348	0.1	402	0.1	428	0.1	445	0.1	441	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`le450-5d` $(450, 5)$																						
k-LF & Best-Fit	283	0.1	324	0.1	359	0.1	394	0.1	427	0.1	444	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-LF & First-Fit	238	0.0	279	0.0	323	0.0	350	0.0	385	0.0	404	0.0	424	0.0	442	0.0	447	0.0	**450**	0.0	**450**	0.0
k-LF & Next-Fit	189	0.0	250	0.0	252	0.0	303	0.0	321	0.0	354	0.0	369	0.0	403	0.0	388	0.0	423	0.0	425	0.0
k-SLF & Best-Fit	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.2	**450**	0.2
k-SLF & First-Fit	227	0.0	260	0.1	343	0.1	361	0.1	414	0.1	430	0.1	449	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
k-SLF & Next-Fit	438	0.1	327	0.1	304	0.1	386	0.1	418	0.1	403	0.1	425	0.1	**450**	0.1	**450**	0.1	**450**	0.1	**450**	0.1
`mulsol.i.1` $(138, 49)$																						
k-LF & Best-Fit	**138**	0.1	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2
k-LF & First-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-LF & Next-Fit	131	0.0	135	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-SLF & Best-Fit	**138**	0.1	**138**	0.2	**138**	0.1	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2	**138**	0.2
k-SLF & First-Fit	**138**	0.1	**138**	0.0	**138**	0.1	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1	**138**	0.1
k-SLF & Next-Fit	137	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
`mulsol.i.2` $(173, 31)$																						
k-LF & Best-Fit	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1
k-LF & First-Fit	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0
k-LF & Next-Fit	166	0.0	172	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0
k-SLF & Best-Fit	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1	**173**	0.1
k-SLF & First-Fit	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0
k-SLF & Next-Fit	172	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0	**173**	0.0
`mulsol.i.3` $(174, 31)$																						
k-LF & Best-Fit	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1
k-LF & First-Fit	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0
k-LF & Next-Fit	168	0.0	173	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0
k-SLF & Best-Fit	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1	**174**	0.1
k-SLF & First-Fit	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.1	**174**	0.0	**174**	0.0	**174**	0.0
k-SLF & Next-Fit	173	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0	**174**	0.0
`mulsol.i.4` $(175, 31)$																						
k-LF & Best-Fit	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1
k-LF & First-Fit	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0
k-LF & Next-Fit	169	0.0	174	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0
k-SLF & Best-Fit	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1	**175**	0.1
k-SLF & First-Fit	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0
k-SLF & Next-Fit	174	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0	**175**	0.0
`mulsol.i.5` $(176, 31)$																						
k-LF & Best-Fit	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1
k-LF & First-Fit	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0
k-LF & Next-Fit	171	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0
k-SLF & Best-Fit	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1	**176**	0.1
k-SLF & First-Fit	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0
k-SLF & Next-Fit	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0	**176**	0.0
`zeroin.i.1` $(126, 49)$																						
k-LF & Best-Fit	**126**	0.1	**126**	0.1	**126**	0.2	**126**	0.1	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2
k-LF & First-Fit	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0
k-LF & Next-Fit	123	0.0	125	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0
k-SLF & Best-Fit	**126**	0.1	**126**	0.1	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2	**126**	0.2
k-SLF & First-Fit	**126**	0.1	**126**	0.1	**126**	0.0	**126**	0.1	**126**	0.1	**126**	0.1	**126**	0.1	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0
k-SLF & Next-Fit	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0	**126**	0.0

Tableau 5.12 Les sommets sont révélés en une itération (suite)

Couleurs k / INSTANCE (n, χ)	χ col.	χ Δt	$\chi+1$ col.	$\chi+1$ Δt	$\chi+2$ col.	$\chi+2$ Δt	$\chi+3$ col.	$\chi+3$ Δt	$\chi+4$ col.	$\chi+4$ Δt	$\chi+5$ col.	$\chi+5$ Δt	$\chi+6$ col.	$\chi+6$ Δt	$\chi+7$ col.	$\chi+7$ Δt	$\chi+8$ col.	$\chi+8$ Δt	$\chi+9$ col.	$\chi+9$ Δt	$\chi+10$ col.	$\chi+10$ Δt
`zeroin.i.2` (157, 30)																						
k-LF & Best-Fit	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1
k-LF & First-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-LF & Next-Fit	148	0.0	155	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-SLF & Best-Fit	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1
k-SLF & First-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-SLF & Next-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
`zeroin.i.3` (157, 30)																						
k-LF & Best-Fit	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1
k-LF & First-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-LF & Next-Fit	148	0.0	155	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-SLF & Best-Fit	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1	**157**	0.1
k-SLF & First-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
k-SLF & Next-Fit	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0	**157**	0.0
`anna` (138, 11)																						
k-LF & Best-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-LF & First-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-LF & Next-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-SLF & Best-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-SLF & First-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
k-SLF & Next-Fit	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0	**138**	0.0
`david` (87, 11)																						
k-LF & Best-Fit	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
k-LF & First-Fit	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
k-LF & Next-Fit	86	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
k-SLF & Best-Fit	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
k-SLF & First-Fit	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
k-SLF & Next-Fit	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0	**87**	0.0
`homer` (556, 13)																						
k-LF & Best-Fit	**556**	0.0	**556**	0.0	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1
k-LF & First-Fit	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0
k-LF & Next-Fit	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0
k-SLF & Best-Fit	**556**	0.0	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1	**556**	0.1
k-SLF & First-Fit	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0
k-SLF & Next-Fit	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0	**556**	0.0
`huck` (74, 11)																						
k-LF & Best-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
k-LF & First-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
k-LF & Next-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
k-SLF & Best-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
k-SLF & First-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
k-SLF & Next-Fit	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0	**74**	0.0
`jean` (77, 10)																						
k-LF & Best-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
k-LF & First-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
k-LF & Next-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
k-SLF & Best-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
k-SLF & First-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
k-SLF & Next-Fit	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0	**77**	0.0
`games120` (120, 9)																						
k-LF & Best-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
k-LF & First-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
k-LF & Next-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
k-SLF & Best-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
k-SLF & First-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
k-SLF & Next-Fit	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0	**120**	0.0
`miles1000` (128, 42)																						
k-LF & Best-Fit	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
k-LF & First-Fit	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-LF & Next-Fit	127	0.0	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Best-Fit	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
k-SLF & First-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Next-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
`miles1500` (128, 73)																						
k-LF & Best-Fit	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3
k-LF & First-Fit	**128**	0.0	**128**	0.1	**128**	0.1	**128**	0.0	**128**	0.1	**128**	0.0	**128**	0.0	**128**	0.1	**128**	0.1	**128**	0.0	**128**	0.1
k-LF & Next-Fit	126	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Best-Fit	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3	**128**	0.3
k-SLF & First-Fit	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
k-SLF & Next-Fit	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
`miles250` (125, 8)																						
k-LF & Best-Fit	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
k-LF & First-Fit	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
k-LF & Next-Fit	123	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
k-SLF & Best-Fit	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
k-SLF & First-Fit	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
k-SLF & Next-Fit	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0	**125**	0.0
`miles500` (128, 20)																						
k-LF & Best-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-LF & First-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-LF & Next-Fit	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Best-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & First-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Next-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0

Tableau 5.12 Les sommets sont révélés en une itération (suite)

Couleurs k	χ		$\chi+1$		$\chi+2$		$\chi+3$		$\chi+4$		$\chi+5$		$\chi+6$		$\chi+7$		$\chi+8$		$\chi+9$		$\chi+10$	
INSTANCE (n,χ)	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt
`miles750` $(128,31)$																						
k-LF & Best-Fit	**128**	0.0	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
k-LF & First-Fit	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-LF & Next-Fit	126	0.0	127	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Best-Fit	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1	**128**	0.1
k-SLF & First-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
k-SLF & Next-Fit	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0	**128**	0.0
`queen11-11` $(121,11)$																						
k-LF & Best-Fit	95	0.0	101	0.0	107	0.0	114	0.0	118	0.0	120	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0
k-LF & First-Fit	92	0.0	100	0.0	106	0.0	111	0.0	116	0.0	119	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0
k-LF & Next-Fit	98	0.0	103	0.0	110	0.0	115	0.0	114	0.0	118	0.0	119	0.0	120	0.0	**121**	0.0	120	0.0	**121**	0.0
k-SLF & Best-Fit	102	0.0	113	0.0	119	0.0	119	0.0	120	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0
k-SLF & First-Fit	102	0.0	109	0.0	116	0.0	119	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0
k-SLF & Next-Fit	100	0.0	111	0.0	117	0.0	118	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0	**121**	0.0
`queen13-13` $(169,13)$																						
k-LF & Best-Fit	132	0.0	142	0.0	151	0.0	157	0.1	163	0.1	166	0.1	**169**	0.1	**169**	0.1	**169**	0.1	**169**	0.1	**169**	0.1
k-LF & First-Fit	124	0.0	132	0.0	140	0.0	147	0.0	153	0.0	157	0.0	160	0.0	163	0.0	166	0.0	168	0.0	**169**	0.0
k-LF & Next-Fit	142	0.0	148	0.0	152	0.0	156	0.0	164	0.0	161	0.0	166	0.0	168	0.0	**169**	0.0	**169**	0.0	**169**	0.0
k-SLF & Best-Fit	143	0.0	152	0.0	162	0.0	165	0.0	167	0.0	**169**	0.1	**169**	0.1	**169**	0.1	**169**	0.1	**169**	0.1	**169**	0.1
k-SLF & First-Fit	144	0.0	153	0.0	165	0.0	168	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0
k-SLF & Next-Fit	147	0.0	153	0.0	164	0.0	**169**	0.0	168	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0	**169**	0.0
`queen5-5` $(25,5)$																						
k-LF & Best-Fit	21	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
k-LF & First-Fit	20	0.0	24	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
k-LF & Next-Fit	**25**	0.0	21	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
k-SLF & Best-Fit	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
k-SLF & First-Fit	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
k-SLF & Next-Fit	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0	**25**	0.0
`queen6-6` $(36,7)$																						
k-LF & Best-Fit	34	0.0	35	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
k-LF & First-Fit	34	0.0	35	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
k-LF & Next-Fit	31	0.0	35	0.0	34	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
k-SLF & Best-Fit	31	0.0	34	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
k-SLF & First-Fit	31	0.0	34	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
k-SLF & Next-Fit	34	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0	**36**	0.0
`queen7-7` $(49,7)$																						
k-LF & Best-Fit	36	0.0	39	0.0	43	0.0	46	0.0	48	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
k-LF & First-Fit	37	0.0	40	0.0	43	0.0	46	0.0	48	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
k-LF & Next-Fit	38	0.0	42	0.0	44	0.0	48	0.0	**49**	0.0	48	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
k-SLF & Best-Fit	38	0.0	42	0.0	47	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
k-SLF & First-Fit	38	0.0	42	0.0	47	0.0	48	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
k-SLF & Next-Fit	41	0.0	45	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0	**49**	0.0
`queen8-12` $(96,12)$																						
k-LF & Best-Fit	84	0.0	89	0.0	94	0.0	95	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
k-LF & First-Fit	85	0.0	89	0.0	92	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
k-LF & Next-Fit	88	0.0	91	0.0	94	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
k-SLF & Best-Fit	93	0.0	95	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
k-SLF & First-Fit	90	0.0	94	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
k-SLF & Next-Fit	92	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0	**96**	0.0
`queen8-8` $(64,9)$																						
k-LF & Best-Fit	54	0.0	57	0.0	60	0.0	62	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
k-LF & First-Fit	52	0.0	56	0.0	60	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
k-LF & Next-Fit	55	0.0	61	0.0	61	0.0	63	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
k-SLF & Best-Fit	58	0.0	61	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
k-SLF & First-Fit	58	0.0	61	0.0	63	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
k-SLF & Next-Fit	60	0.0	61	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0	**64**	0.0
`queen9-9` $(81,10)$																						
k-LF & Best-Fit	67	0.0	72	0.0	77	0.0	77	0.0	79	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
k-LF & First-Fit	64	0.0	69	0.0	74	0.0	78	0.0	80	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
k-LF & Next-Fit	69	0.0	76	0.0	75	0.0	80	0.0	**81**	0.0	77	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
k-SLF & Best-Fit	69	0.0	76	0.0	80	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
k-SLF & First-Fit	72	0.0	75	0.0	78	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
k-SLF & Next-Fit	71	0.0	77	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0	**81**	0.0
`myciel3` $(11,4)$																						
k-LF & Best-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
k-LF & First-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
k-LF & Next-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
k-SLF & Best-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
k-SLF & First-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
k-SLF & Next-Fit	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0	**11**	0.0
`myciel4` $(23,5)$																						
k-LF & Best-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
k-LF & First-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
k-LF & Next-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
k-SLF & Best-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
k-SLF & First-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0
k-SLF & Next-Fit	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0	**23**	0.0

Tableau 5.12 Les sommets sont révélés en une itération (suite et fin)

Couleurs k	χ		$\chi+1$		$\chi+2$		$\chi+3$		$\chi+4$		$\chi+5$		$\chi+6$		$\chi+7$		$\chi+8$		$\chi+9$		$\chi+10$	
INSTANCE (n,χ)	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt	col.	Δt
`myciel5` $(47,6)$																						
k-LF & Best-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
k-LF & First-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
k-LF & Next-Fit	46	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
k-SLF & Best-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
k-SLF & First-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
k-SLF & Next-Fit	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0	**47**	0.0
`myciel6` $(95,7)$																						
k-LF & Best-Fit	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
k-LF & First-Fit	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
k-LF & Next-Fit	92	0.0	94	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
k-SLF & Best-Fit	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
k-SLF & First-Fit	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
k-SLF & Next-Fit	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0	**95**	0.0
`myciel7` $(191,8)$																						
k-LF & Best-Fit	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0
k-LF & First-Fit	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0
k-LF & Next-Fit	190	0.0	183	0.0	190	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0
k-SLF & Best-Fit	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.1
k-SLF & First-Fit	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0
k-SLF & Next-Fit	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0	**191**	0.0

CHAPITRE 6 CONCLUSION

Nous concluons cette thèse en récapitulant brièvement les principaux résultats, ainsi que leurs limites, qui laissent naturellement place à de nouveaux axes de recherche possibles.

6.1 Synthèse des travaux

Nous avons traité le problème combinatoire suivant : comment assigner un canal de communication à un maximum d'usagers, de façon à ce que la contrainte suivante soit respectée pour tout usager i connecté à sa station p via le canal $c(i)$:

$$\sum_{j \neq i | c(j)=c(i)} P_{jp} + \mathcal{P}_0 \leq \theta P_{ip},$$

ou en termes de graphes, comment colorer un maximum de sommets, sans violer l'inégalité suivante :

$$\sum_{j \neq i | c(j)=c(i)} \omega(j,i) \leq \theta W(i).$$

Ce problème étant $\mathcal{NP}$-difficile, deux approches ont été développées pour le résoudre : une approche exacte basée sur la programmation linéaire en nombre entiers, et une approche heuristique basée sur les algorithmes constructifs de coloration usuels. Nous avons ensuite étendu le modèle sur un horizon de temps, où les données du problème apparaissent de façon dynamique.

Résolution exacte en nombres entiers

Dans un premier temps, nous avons développé des formulations mathématiques à l'aide de variables binaires pour modéliser le problème. Ces modèles peuvent être directement résolus par des solveurs si le nombre de sommets du graphe en question n'est pas trop élevé, et nous avons montré qu'il est possible d'accélérer le temps de résolution en ajoutant des inégalités valides, des bornes supérieures sur l'objectif, ou encore en procédant à des changements de variables donnant lieu à des formulations dites de décomposition.

Bien que cette approche ne soit pas utilisable en pratique pour des affectations en temps réel, elle permet de bien comprendre l'aspect combinatoire du problème. De plus, l'accès à la solution optimale pour de petites instances sert de référence pour les heuristiques développées dans le Chapitre 4.

Résolution approchée par heuristiques

L'approche en nombres entiers s'avérant non utilisable en pratique, le choix de développer des heuristiques s'est rapidement imposé. Les méthodes proposées sont des généralisations des heuristiques constructives usuelles pour la coloration des sommets d'un graphe, à savoir Largest First, Saturation Largest First et Recursive Largest First. Celles-ci ont été améliorées en exploitant la structure sous-jacente aux réseaux de téléphonie mobile, au moyen de pavages de Voronoï notamment. Nous avons étudié la complexité de ces algorithmes ainsi que les plus petites instances sous-optimales. Ces deux caractéristiques sont souvent utilisées pour évaluer la performance d'une heuristique qui, par nature, ne peut garantir de converger vers une solution optimale.

Afin de valider notre modèle et nos heuristiques, nous les avons comparées dans un premier temps aux solutions optimales obtenues au Chapitre 3, puis aux techniques utilisées actuellement par les opérateurs téléphoniques, à savoir la réutilisation de fréquence fractionnée. Nous avons déterminé une borne supérieure sur le nombre maximum d'usagers pouvant être simultanément connectés au réseau et avons montré numériquement que nos heuristiques font quasi-systématiquement mieux, autrement dit le modèle proposé ici permet d'utiliser les ressources du réseau de façon plus efficace.

Extension du modèle sur un horizon de temps

Enfin, ce modèle a été étendu sur un horizon de temps, où les usagers (ou les sommets du graphe) apparaissent itérativement un par un, ou éventuellement par paquets. Nous avons étudié les meilleures garanties possibles pour ces configurations, en particulier nous avons montré qu'aucun algorithme ne peut systématiquement garantir une solution de taille (en nombre de sommets) supérieure ou égale à $\frac{k}{\sqrt{n}-1}$ fois la solution optimale. Nous avons alors repris les algorithmes du Chapitre 4 et les avons adaptés à ces scénarios, notamment en étudiant différentes politiques de priorité. Des tests numériques basés sur les intances DIMACS nous ont permis de valider ces heuristiques.

Enfin, nous avons relâché certaines contraintes du problème, en permettant de revenir sur certaines décisions du passé, moyennant un certain coût. Ceci donne lieu à un problème bi-objectif, que nous avons traité avec deux approches différentes : l'une basée sur des algorithmes dits de réparation, l'autre sur des méthodes de recherche locale. Bien que les deux approches présentent des avantages et des inconvénients, la recherche locale s'avère plus intéressante pour la plupart des cas.

6.2 Limitations de la solution proposée et améliorations futures

Tel qu'expliqué en introduction, tout problème de recherche opérationnelle passe par une phase de modélisation, et une autre de résolution, souvent à l'aide d'algorithmes. Il est certainement possible d'améliorer ces deux phases.

Limites et améliorations du modèle

La modélisation est une tâche complexe pour au moins deux raisons. D'une part, être le plus fidèle possible à la réalité sans compliquer excessivement le modèle est un juste milieu subtil à déterminer. D'autre part, le degré de difficulté de résolution varie d'un modèle à l'autre (même s'ils représentent la réalité avec la même précision).

Nous avons montré en quoi le modèle de coloration de graphes permet bel et bien de représenter le problème d'affectation de canaux, cependant nous nous sommes uniquement intéressés à la notion de *connectivité*, et non au *débit* de transfert de données. En réalité, un groupe de porteuses (et non pas une seule) est affecté à un usager, et la capacité de transfert des données est directement liée à la distribution de ces groupes. Clairement, cet aspect n'a pas été abordé ici. Cependant, il serait possible de généraliser notre modèle pour le prendre en compte de la façon suivante : plutôt que d'affecter une seule couleur à chaque sommet, il faudrait en affecter plusieurs. Ceci rentre dans le cadre de la multi-coloration des sommets, notion déjà existante dans la littérature.

Une des difficultés encourues lors du développement du Chapitre 4 était de pouvoir comparer le modèle proposé ici avec celui des fournisseurs de services mobiles actuels, dont les algorithmes d'affectation ne sont pas publics. Nous nous sommes donc basés sur le modèle sous-jacent à ces algorithmes (la réutilisation de fréquence fractionnée) et avons dérivé une borne supérieure sur ce que celui-ci permet :

$$\sum_{i=1}^{t} \min\{x_i, k\} + \min\left\{y_i, \frac{k}{3}, \max\{0, k - x_i\}\right\}. \tag{6.1}$$

Cependant, nous avons vu en introduction que de nombreuses méthodes d'affectations statiques permettent aux cellules d'emprunter des canaux aux cellules voisines, de façon à prendre en compte les variations de trafic au cours du temps. Cette borne supérieure ne considère pas cette possibilité ; si l'on pouvait montrer que le modèle proposé ici (couplé aux heuristiques) fait mieux, son efficacité serait encore mieux démontrée. Il est possible de déterminer une telle borne, en faisant à nouveau appel au graphe de la Proposition 18. Si l'échange de canaux est permis entre deux cellules adjacentes, c'est qu'il concerne unique-

ment les zones en bordure (puisque les zones centrales ont déjà accès à l'ensemble du spectre disponible). Ainsi, si l'on veut permettre un tel échange entre deux cellules C_1 et C_2, il suffit de relier les deux sommets par lesquels entrent les flots y_1 et y_2 par deux arcs, un dans chaque sens. Grâce à ce modèle, il serait possible d'effectuer plus de tests numériques afin d'affiner la comparaison entre les heuristiques proposées et la réutilisation de fréquence fractionnée actuellement utilisée par les opérateurs.

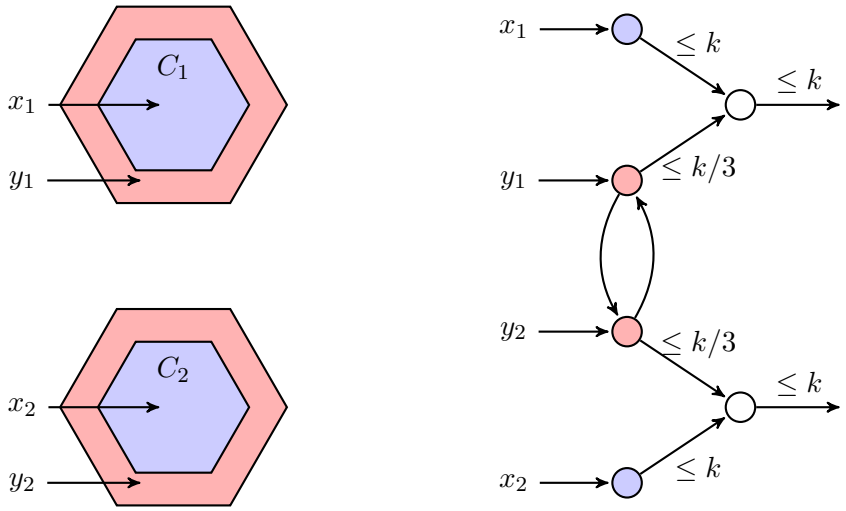


Figure 6.1 Détermination par un flot maximum du nombre maximum d'usagers connectés dans deux cellules pouvant s'échanger des canaux

Enfin, l'incertitude liée à l'apparition et la disparition des usagers dans le réseau a été traitée par une approche algorithmique on-line. D'autre voies telles que l'optimisation stochastique sont possibles, et il serait intéressant de comparer ces deux approches. De plus, de nombreux modèles décrivant le trafic dans un réseau de téléphonie mobile (spatial et temporel) existent, et il est clair que l'on pourrait effectuer de nombreux autres tests numériques pour évaluer le comportement de nos algorithmes dans ces configurations.

Limites et améliorations des algorithmes

Concernant les algorithmes, de nombreuses voies d'améliorations sont également envisageables. Premièrement, le modèle proposé ici s'appuie sur une centralisation des données, nous avons donc naturellement développé des algorithmes en tenant compte de cette hypothèse. Cependant, dans les faits, les opérateurs ne procèdent pas de cette façon, les standards de quatrième génération (4G) tels que le standard LTE ne pourraient donc pas utiliser les heuristiques proposées ici telles quelles.

On peut toutefois espérer que les standards à venir (5G) pourront intégrer ce modèle. Si ce n'est pas le cas, il existe de nombreux algorithmes à structure décentralisée qui pourraient le résoudre sans avoir une vue globale du réseau. Il serait très intéressant d'évaluer d'une part le coût d'installation et d'utilisation d'une entité en mesure de centraliser les données à l'échelle métropolitaine, d'autre part de comparer les heuristiques de cette thèse avec des heuristiques de coloration décentralisées.

De plus, nous avons adapté les heuristiques Largest First, Saturation Largest First, et Recursive Largest First, mais il existe d'autres algorithmes constructifs usuels, tel que mentionné au Chapitre 2. En particulier, l'algorithme Smallest Last, également basé sur la notion de degré d'un graphe, serait facile à mettre en oeuvre. Il serait intéressant de le comparer numériquement aux autres. Concernant les méthodes exactes, l'algorithme branch-and-price utilisé s'appuie sur une décomposition de Dantzig du problème, résolu par un procédé de génération de colonnes. Au cours de ce procédé, on obtient une borne inférieure sur l'objectif. Nous avons vu le lien avec la relaxation Lagrangienne mais ne l'avons pas exploité numériquement. Celle-ci pourrait éventuellement accélérer le parcours de l'arbre de branchement commun à ces méthodes de résolution en nombres entiers. De même nous avons mentionné des stratégies de branchement alternatives, mais ne les avons pas implantées. Il serait intéressant de voir dans quelle mesure elles permettent de diminuer les temps de calculs.

Le seul élément de structure utilisé dans ces travaux, via les diagrammes de Voronoï, est la position géographique des stations de base dans le plan. Il serait particulièrement pertinent de développer l'approche de Araujo et al. [46], qui ont entièrement caractérisé le nombre chromatique $\chi_\theta(G^2, \omega)$ sur des classes particulières de graphes (les arbres et les grilles) souvent utilisées en télécommunications (en l'occurence il s'agissait d'affectation de fréquences pour un réseau de satellites).

Enfin, concernant les algorithmes on-line, nous avons considéré des algorithmes de réparation et de recherche locale. Il est probable qu'un algorithme hybride capable de faire de la recherche locale et de reconstruire les classes de couleur selon le niveau de saturation du réseau soit encore plus performant.

Axe de recherche supplémentaire

Cette thèse porte (entre autres) sur une version pondérée et orientée du problème de la coloration θ-impropre. Si l'on oublie les pondérations, l'orientation, et que l'on fixe θ à une valeur entière, une question naturelle se pose : combien de couleurs peut-on éliminer si l'on autorise θ conflits par sommets ? Existe-t-il des graphes pour lesquels aucune couleur ne peut être éliminée ? Si oui, comment caractériser ces graphes ? Nous proposons quelques éléments

de réponses à ces questions qui méritent d'être approfondies.

Il n'est pas difficile de voir qu'un tel graphe doit nécessairement vérifier la condition suivante :

$$n \geq (\theta + 1)\chi - \theta.$$

En effet, montrons que l'hypothèse inverse mène à une contradiction, supposons donc qu'il existe un graphe qui vérifie $\chi = \chi_\theta$ et $n < (\theta+1)\chi-\theta$. D'après l'inégalité de Lovász introduite au Chapitre 2, on a

$$\chi_\theta \leq \left\lceil \frac{\Delta + 1}{\theta + 1} \right\rceil,$$

et puisque $\Delta \leq n - 1$, il s'ensuit que

$$\chi_\theta \leq \left\lceil \frac{n}{\theta + 1} \right\rceil.$$

Or, d'après notre seconde hypothèse, $n < (\theta + 1)\chi - \theta$, d'où

$$\chi_\theta < \left\lceil \chi - \frac{\theta}{\theta + 1} \right\rceil = \chi,$$

ce qui contredit effectivement la première hypothèse, $\chi = \chi_\theta$.

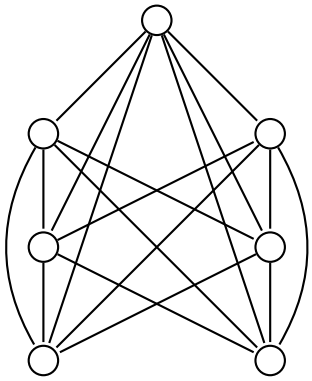


Figure 6.2 Un graphe qui vérifie $\chi = \chi_1 = 4$

Peut-on exhiber un tel graphe ? Voyons le cas particulier où $\theta = 1$, et considérons le graphe $G_k = K_{2k-1} - M_{k-1}$ composé d'une clique d'ordre $2k - 1$ à laquelle on a retiré un couplage d'ordre $k - 1$ (un exemple est illustré à la Figure 6.2 avec $k = 4$). Tout d'abord, ce graphe est bien candidat pour vérifier $\chi_1 = \chi(= k)$, puisqu'il satisfait bien la condition nécessaire ci-dessus. De plus, toutes ses k-colorations ont nécessairement des classes de couleurs de cardinalités $1, 2, 2, \cdots, 2$. Et si l'on ôte encore un couplage quelconque M du graphe, on ne

pourra jamais créer de triangle de non-arêtes, donc le nombre de stabilité du graphe résultant restera égal à 2. Ceci signifie que

$$\chi_1 = \chi(G_k - M) \geq \left\lceil \frac{n}{\alpha(G_k - M)} \right\rceil = \left\lceil \frac{2k-1}{2} \right\rceil = k = \chi,$$

donc G_k est un exemple de graphe pour lequel l'autorisation de $\theta = 1$ conflit par sommet ne permet pas d'éliminer de couleurs.

Est-ce le plus petit graphe qui vérifie cela pour $\chi = k$ fixé ? G_k possède un total de $n = 2k-1$ sommets et $m = 2(k-1)^2$ arêtes. Nous laissons le soin au lecteur de réfléchir à la conjecture suivante.

Conjecture : Si un graphe G vérifie $\chi = \chi_1$, alors

$$m \geq 2(\chi - 1)^2.$$

Le mot de la fin

J'espère, par ces travaux, avoir apporté une petite goutte d'eau à l'océan des connaissances scientifiques, une petite brique à l'édifice dont les fondations sont dues entre autres à Leonhard Euler, Claude Berge, George Dantzig, Maurice Manori (voir Figure 6.3 ci-après), et bien d'autres...

Figure 6.3 Leonhard Euler (1707-1783), père fondateur de la théorie des graphes ; George Dantzig (1914-2005), inventeur de l'algorithme du simplexe ; Claude Berge (1926-2002), grand théoricien des graphes et fondateur de l'Oulipo ; Maurice Manori (1962-), spécialiste mondial de la coloration de graphes, détective [2].

2. Sources :
http://fr.wikipedia.org/wiki/Leonhard_Euler,
http://www.the1thing.com/blog/applying-the-one-thing/solving-the-unsolvable/,
http://moc.g-scop.grenoble-inp.fr/?p=5,
http://www.polymtl.ca/pub/sites/lagrapheur/

RÉFÉRENCES

[1] Ericsson. (2013) Ericsson mobility report - on the pulse of networked society. [Online]. Available : http://www.ericsson.com/res/docs/2013/ericsson-mobility-report-june-2013.pdf

[2] Idate. (2013) Digiworld yearbook - understanding the digital world. [Online]. Available : http://www.itrpress.com/cp/2013/2013-06-10_DigiWorld-Yearbook-2013-Dossier-de-presse.pdf

[3] C. Archetti, N. Bianchessi, A. Hertz, A. Colombet, and F. Gagnon, "Directed weighted improper coloring for cellular channel allocation," *Discrete Applied Mathematics*, 2013. [Online]. Available : http://dx.doi.org/10.1016/j.dam.2013.11.018

[4] T. S. Rappaport *et al.*, *Wireless communications : principles and practice*. Prentice Hall PTR New Jersey, 1996, vol. 2.

[5] P. Kyosti, J. Meinila, L. Hentila, and X. Zhao, "WINNER II channel models : Part I channel models version 1.2," *WINNER and Information Society Technologies," Technical Report*, 2007.

[6] É. polytechnique de Montréal. Département de mathématiques et de génie industriel and P. M. Adjakplé, *Affectation de canaux dans les réseaux de téléphonie mobile cellulaire*, 1998.

[7] U. Gotzner and R. Rathgeber, "Spatial traffic distribution in cellular networks," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, vol. 3. IEEE, 1998, pp. 1994–1998.

[8] F. S. Roberts, "T-colorings of graphs : recent results and open problems," *Discrete mathematics*, vol. 93, no. 2, pp. 229–245, 1991.

[9] I. Baybars, "Optimal assignment of broadcasting frequencies," *European Journal of Operational Research*, vol. 9, no. 3, pp. 257–263, 1982.

[10] A. Giortzis and L. Turner, "Application of mathematical programming to the fixed channel assignment problem in mobile radio networks," *IEE Proceedings-Communications*, vol. 144, no. 4, pp. 257–264, 1997.

[11] B. Jaumard, O. Marcotte, C. Meyer, and Q. Groupe d'études et de recherche en analyse des décisions (Montréal, *Estimation of the quality of cellular networks using column generation techniques*. Citeseer, 1998.

[12] B. Jaumard, O. Marcotte, C. Meyer, and T. Vovor, "Erratum to "comparison of column generation models for channel assignment in cellular networks" :[discrete appl. math. 112 (2001) 217–240]," *Discrete Applied Mathematics*, vol. 118, no. 3, pp. 299–322, 2002.

[13] R. Mathar and J. Mattfeldt, "Channel assignment in cellular radio networks," *Vehicular Technology, IEEE Transactions on*, vol. 42, no. 4, pp. 647–656, 1993.

[14] K.-N. Chang and S. Kim, "Channel allocation in cellular radio networks," *Computers & operations research*, vol. 24, no. 9, pp. 849–860, 1997.

[15] G. Verfaillie, M. Lemaître, and T. Schiex, "Russian doll search for solving constraint optimization problems," in *AAAI/IAAI, Vol. 1.* Citeseer, 1996, pp. 181–187.

[16] J. P. Warners, T. Terlaky, C. Roos, and B. Jansen, "A potential reduction approach to the frequency assignment problem," *Discrete Applied Mathematics*, vol. 78, no. 1, pp. 251–282, 1997.

[17] M. Padberg, "The boolean quadric polytope : some characteristics, facets and relatives," *Mathematical programming*, vol. 45, no. 1-3, pp. 139–172, 1989.

[18] A. M. Koster, S. P. Van Hoesel, and A. W. Kolen, "The partial constraint satisfaction problem : Facets and lifting theorems," *Operations research letters*, vol. 23, no. 3, pp. 89–97, 1998.

[19] S. Tiourine, C. Hurkens, and J. K. Lenstra, "An overview of algorithmic approaches to frequency assignment problems," in *Proceedings of CALMA Symposium, Scheveningen*, 1995.

[20] A. Eisenblätter *et al.*, *Frequency assignment in GSM networks : Models, heuristics, and lower bounds.* Cuvillier, 2001.

[21] K. I. Aardal, S. P. Van Hoesel, A. M. Koster, C. Mannino, and A. Sassano, "Models and solution techniques for frequency assignment problems," *Annals of Operations Research*, vol. 153, no. 1, pp. 79–129, 2007.

[22] L. G. Anderson, "A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system," *Vehicular Technology, IEEE Transactions on*, vol. 22, no. 4, pp. 210–217, 1973.

[23] J. S. Engel and M. M. Peritsky, "Statistically-optimum dynamic server assignment in systems with interfering servers," *Communications, IEEE Transactions on*, vol. 21, no. 11, pp. 1287–1293, 1973.

[24] T. Kahwa and N. Georganas, "A hybrid channel assignment scheme in large-scale, cellular-structured mobile communication systems," *Communications, IEEE Transactions on*, vol. 26, no. 4, pp. 432–438, 1978.

[25] M. Zhang and T.-S. P. Yum, "Comparisons of channel-assignment strategies in cellular mobile telephone systems," *Vehicular Technology, IEEE Transactions on*, vol. 38, no. 4, pp. 211–215, 1989.

[26] S. M. Elnoubi, R. Singh, and S. C. Gupta, "A new frequency channel assignment algorithm in high capacity mobile communication systems," *Vehicular Technology, IEEE Transactions on*, vol. 31, no. 3, pp. 125–131, 1982.

[27] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems : A comprehensive survey," *Personal Communications, IEEE*, vol. 3, no. 3, pp. 10–31, 1996.

[28] D. C. Cox and D. O. Reudink, "Dynamic channel assignment in two-dimensional large-scale mobile radio systems," *Bell System Technical Journal*, vol. 51, no. 7, pp. 1611–1629, 1972.

[29] K. Okada and F. Kubota, "On dynamic channel assignment strategies in cellular mobile radio systems," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 75, no. 12, pp. 1634–1641, 1992.

[30] D. C. Cox and D. O. Reudink, "A comparison of some channel assignment strategies in large-scale mobile communications systems," *Communications, IEEE Transactions on*, vol. 20, no. 2, pp. 190–195, 1972.

[31] I. Chih-Lin and P.-H. Chao, "Local packing-distributed dynamic channel allocation at cellular base station," in *Global Telecommunications Conference, 1993, including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston. GLOBECOM'93., IEEE.* IEEE, 1993, pp. 293–301.

[32] C. Lin, P.-H. Chao *et al.*, "Distributed dynamic channel allocation algorithms with adjacent channel constraints," in *Personal, Indoor and Mobile Radio Communications, 1994. Wireless Networks-Catching the Mobile Future., 5th IEEE International Symposium on*, vol. 1. IEEE, 1994, pp. 169–177.

[33] M. Serizawa and D. Goodman, "Instability and deadlock of distributed dynamic channel allocation," in *Vehicular Technology Conference, 1993., 43rd IEEE.* IEEE, 1993, pp. 528–531.

[34] J. B. Punt, D. Sparreboom, and F. Brouwer, "Mathematical models for the analysis of dynamic channel selection for indoor mobile wireless communication systems," in *Personal, Indoor and Mobile Radio Communications, 1994. Wireless Networks-Catching the Mobile Future., 5th IEEE International Symposium on*, vol. 4. IEEE, 1994, pp. 1081–1085.

[35] D. Hong and S. Rappaport Stephen, "Traffic model and performance analysis for cellular mobile radio telephone systems with prioritized and nonprioritized handoff procedures," *Vehicular Technology, IEEE Transactions on*, vol. 35, no. 3, pp. 77–92, 1986.

[36] Y. Furuya and Y. Akaiwa, "Channel segregation, a distributed adaptive channel allocation scheme for mobile communication systems," *IEICE Transactions on Communications*, vol. 74, no. 6, pp. 1531–1537, 1991.

[37] Y. Akaiwa and H. Andoh, "Channel segregation-a self-organized dynamic channel allocation method : application to tdma/fdma microcellular system," *Selected Areas in Communications, IEEE Journal on*, vol. 11, no. 6, pp. 949–954, 1993.

[38] K. Appel, W. Haken *et al.*, "Every planar map is four colorable. part i : Discharging," *Illinois Journal of Mathematics*, vol. 21, no. 3, pp. 429–490, 1977.

[39] D. de Werra, "An introduction to timetabling," *European Journal of Operational Research*, vol. 19, no. 2, pp. 151–162, 1985.

[40] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[41] P. Erdős and A. Hajnal, "On decomposition of graphs," *Acta Mathematica Hungarica*, vol. 18, no. 3, pp. 359–377, 1967.

[42] J. A. Andrews and M. S. Jacobson, "On a generalization of chromatic number," *Congressus Numerantium*, vol. 47, pp. 33–48, 1985.

[43] F. Harary, "Conditional colorability in graphs." Graphs and Applications, Proc. First Colo. Symp. graph theory (F. Harary and J. Maybee eds), Wiley intersci., Publ. NY, 1985.

[44] L. J. Cowen, R. Cowen, and D. Woodall, "Defective colorings of graphs in surfaces : partitions into subgraphs of bounded valency," *Journal of Graph Theory*, vol. 10, no. 2, pp. 187–195, 1986.

[45] K. Appel and W. Haken, "Solution of the four color map problem," *Scientific American*, vol. 237, no. 4, pp. 108–121, 1977.

[46] J. Araujo, J.-C. Bermond, F. Giroire, F. Havet, D. Mazauric, and R. Modrzejewski, "Weighted improper colouring," in *Proceedings of the 22nd international conference on Combinatorial Algorithms*, ser. IWOCA'11. Berlin, Heidelberg : Springer-Verlag, 2011, pp. 1–18. [Online]. Available : http://dx.doi.org/10.1007/978-3-642-25011-8_1

[47] H. Tamura, M. Sengoku, S. Shinoda, and A. Takeo, "Channel assignment problem in a cellular mobile system and a new coloring problem of networks," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 74, no. 10, pp. 2983–2989, 1991.

[48] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 3, pp. 19–31, 2005.

[49] R. M. Karp, *Reducibility among combinatorial problems.* Springer, 1972.

[50] R. Lewis, *A Guide to Graph Colouring : Algorithms and Applications.* Springer, 2015.

[51] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and approximation : Combinatorial optimization problems and their approximability properties.* Springer Science & Business Media, 2012.

[52] M. R. Garey and D. S. Johnson, "Computers and intractability : a guide to the theory of np-completeness. 1979," *San Francisco, LA : Freeman*, 1979.

[53] G. Narasimhan *et al.*, "The maximum k-colorable subgraph problem," Ph.D. dissertation, University of Wisconsin, Madison, 1989.

[54] W. Imrich, S. Klavzar, and D. F. Rall, *Topics in graph theory : Graphs and their Cartesian product.* AK Peters Ltd, 2008.

[55] L. A. Wolsey, *Integer programming.* Wiley New York, 1998, vol. 42.

[56] G. B. Dantzig, *Origins of the simplex method.* ACM, 1990.

[57] P. Pudlák, "Lower bounds for resolution and cutting plane proofs and monotone computations," *The Journal of Symbolic Logic*, vol. 62, pp. 981–998, 9 1997. [Online]. Available : http://journals.cambridge.org/article_S0022481200016133

[58] A. Mehrotra and M. A. Trick, "A column generation approach for graph coloring," *informs Journal on Computing*, vol. 8, no. 4, pp. 344–354, 1996.

[59] B. Myers and R.-w. Liu, "A lower bound on the chromatic number of a graph," *Networks*, vol. 1, no. 3, pp. 273–277, 1971.

[60] N. Brauner and M. Gabay, "Laboratoire g-scop, coloration optimale de graphes," 2010.

[61] J. Bang-Jensen and M. Halldórsson, "A note on vertex coloring edge-weighted digraphs," *Preprints on graph, hypergraphs and computing, Institute Mittag-Leffler*, 2014.

[62] B. A. Gudmundsson, T. K. Magnusson, and B. O. Saemundsson, "Bounds and fixed-parameter algorithms for weighted improper coloring."

[63] D. E. Knuth, *The sandwich theorem.* Stanford University, Department of Computer Science, 1993.

[64] N. Patra, B. B. Ray, and S. P. Mohanty, "Dynamic interval graph coloring for channel assignment in wireless cellular networks."

[65] R. J. Kang, "Improper colourings of graphs," Ph.D. dissertation, University of Oxford, 2008.

[66] F. Aurenhammer and R. Klein, "Voronoi diagrams," *Handbook of computational geometry*, vol. 5, pp. 201–290, 2000.

[67] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[68] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Journal of the ACM (JACM)*, vol. 30, no. 3, pp. 417–427, 1983.

[69] A. Hertz and D. de Werra, "Connected sequential colorings," *Discrete mathematics*, vol. 74, no. 1, pp. 51–59, 1989.

[70] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[71] D. S. Johnson, "Worst case behavior of graph coloring algorithms," in *Proc. 5th SE Conf. on Combinatorics, Graph Theory and Computing*, 1974, pp. 513–528.

[72] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of research of the national bureau of standards*, vol. 84, no. 6, pp. 489–506, 1979.

[73] A. Kosowski and K. Manuszewski, "Classical coloring of graphs," *Contemporary Mathematics*, vol. 352, pp. 1–20, 2004.

[74] M. Bellaıche, "A new efficient rlf-like algorithm for the vertex coloring problem m. adegbindin, a. hertz," 2014.

[75] M. M. Halldórsson, "A still better performance guarantee for approximate graph coloring," *Information Processing Letters*, vol. 45, no. 1, pp. 19–23, 1993.

[76] A. Wigderson, "Improving the performance guarantee for approximate graph coloring," *Journal of the ACM (JACM)*, vol. 30, no. 4, pp. 729–735, 1983.

[77] B. Berger and J. Rompel, "A better performance guarantee for approximate graph coloring," *Algorithmica*, vol. 5, no. 1-4, pp. 459–466, 1990.

[78] G. Palubeckis, "On the recursive largest first algorithm for graph colouring," *International Journal of Computer Mathematics*, vol. 85, no. 2, pp. 191–200, 2008.

[79] P. Hansen and J. Kuplinsky, "The smallest hard-to-color graph," *Discrete mathematics*, vol. 96, no. 3, pp. 199–212, 1991.

[80] M. Kubale and K. Manuszewski, "The smallest hard-to-color graphs for the classical, total and strong colorings of vertices," *Control and Cybernetics*, vol. 28, no. 2, pp. 355–365, 1999.

[81] L. Babel and G. Tinhofer, "Hard-to-color graphs for connected sequential colorings," *Discrete Applied Mathematics*, vol. 51, no. 1, pp. 3–25, 1994.

[82] F. Glover and M. Laguna, *Tabu Search*. Springer, 2013.

[83] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, no. 4, pp. 345–351, 1987.

[84] C. Friden, A. Hertz, and D. de Werra, "Stabulus : A technique for finding stable sets in large graphs with tabu search," *Computing*, vol. 42, no. 1, pp. 35–44, 1989.

[85] A. Gyárfás and J. Lehel, "On-line and first fit colorings of graphs," *Journal of Graph theory*, vol. 12, no. 2, pp. 217–227, 1988.

[86] H. A. Kierstead, S. G. Penrice, and W. T. Trotter, "On-line and first-fit coloring of graphs that do not induce p_5," *SIAM Journal on Discrete Mathematics*, vol. 8, no. 4, pp. 485–498, 1995.

[87] H. A. Kierstead and W. T. Trotter, "An extremal problem in recursive combinatorics," *Congressus Numerantium*, vol. 33, no. 143-153, pp. 13–32, 1981.

[88] M. Slusarek, "Optimal on-line coloring of circular arc graphs," *Informatique théorique et applications*, vol. 29, no. 5, pp. 423–429, 1995.

[89] L. Ouerfelli and H. Bouziri, "Greedy algorithms for dynamic graph coloring," in *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, 2011.

[90] B. Escoffier, "Problème on-line du stable de cardinalité maximale," *Mémoire de DEA*, 2002.

[91] I. SET, "On-line models and algorithms for max," *ANNALES DU LAMSADE N 2 Juin 2004*, p. 219, 2004.

[92] L. M. Favrholdt and J. W. Mikkelsen, "Online dual edge coloring of paths and trees," in *Approximation and Online Algorithms*. Springer, 2014, pp. 181–192.

[93] F. Harary and G. Gupta, "Dynamic graph models," *Mathematical and Computer Modelling*, vol. 25, no. 7, pp. 79–87, 1997.

[94] D. Preuveneers and Y. Berbers, "Acodygra : an agent algorithm for coloring dynamic graphs," *Symbolic and Numeric Algorithms for Scientific Computing (September 2004)*, vol. 6, pp. 381–390, 2004.

[95] A. Dupont, A. C. Linhares, C. Artigues, D. Feillet, P. Michelon, and M. Vasquez, "The dynamic frequency assignment problem," *European Journal of Operational Research*, vol. 195, no. 1, pp. 75–88, 2009.

[96] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[97] K. Kaparis and A. N. Letchford, "Separation algorithms for 0-1 knapsack polytopes," *Mathematical programming*, vol. 124, no. 1-2, pp. 69–91, 2010.

[98] M. O'Sullivan, Q. Lim, C. Walker, I. Dunning, and S. Mitchell, *Dippy : A Simplified Interface for Advanced Mixed-integer Programming*. Citeseer, 2012.

[99] L. Lovász, "Stable sets and polynomials," *Discrete mathematics*, vol. 124, no. 1, pp. 137–153, 1994.

[100] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*. Springer Science & Business Media, 2012, vol. 2.

[101] S. Rebennack, "Stable set problem : Branch & cut algorithms stable set problem : Branch & cut algorithms," in *Encyclopedia of Optimization*. Springer, 2009, pp. 3676–3688.

[102] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," DTIC Document, Tech. Rep., 1988.

[103] N. Robertson, D. Sanders, P. Seymou, and T. R, "The four-colour theorem," *Journal of Combinatorial Theory, Series B*, vol. 70, pp. 2–44, 1967.

[104] R. Ullah, N. Fisal, H. Safdar, W. Maqbool, Z. Khalid, and A. S. Khan, "Voronoi cell geometry based dynamic fractional frequency reuse for ofdma cellular networks," in *Signal and Image Processing Applications (ICSIPA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 435–440.

[105] G. Boudeau, J. Panicker, N. Guo, R. Chang, N. Wang, and S. Vrzic, "Interference coordination and cancellation for 4g networks," *IEEE Communications Magazine*, vol. 47, no. 4, pp. 74–81, 2009.

[106] R. Y. Chang, Z. Tao, J. Zhang, and C. J. Kuo, "A graph approach to dynamic fractional frequency reuse (ffr) in multi-cell ofdma networks," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[107] L. Chen and D. Yuan, "Generalizing and optimizing fractional frequency reuse in broadband cellular radio access networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–15, 2012.

[108] T. Novlan, J. G. Andrews, I. Sohn, R. K. Ganti, and A. Ghosh, "Comparison of fractional frequency reuse approaches in the ofdma cellular downlink," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–5.

[109] P. Kyosti, J. Meinila, L. Hentila, and X. Zhao, "Winner ii channel models : Part i channel models version 1.2," *WINNER and Information Society Technologies," Technical Report*, 2007.

[110] A. Ghosh, J. Zhang, J. G. Andrews, and R. Muhamed, *Fundamentals of LTE*. Pearson Education, 2010, ch. 5, p. 153.

[111] M. A. Zafer, "Channel assignment algorithms and blocking probability analysis for connection-oriented traffic in wireless networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[112] M. Gudmundson, "Analysis of handover algorithms [microcellular radio]," in *Vehicular Technology Conference, 1991. Gateway to the Future Technology in Motion., 41st IEEE.* IEEE, 1991, pp. 537–542.

[113] N. Ekiz, T. Salih, S. Kucukoner, and K. Fidanboylu, "An overview of handoff techniques in cellular networks," *International journal of information technology*, vol. 2, no. 3, pp. 132–136, 2005.

[114] D. Everitt and N. Macfadyen, "Analysis of multicellular mobile radiotelephone systems with loss," *British Telecom Technology Journal*, vol. 1, no. 2, pp. 37–45, 1983.

[115] P. Sankowski, "Weighted bipartite matching in matrix multiplication time," in *Automata, Languages and Programming*. Springer, 2006, pp. 274–285.