UNIVERSITÉ DE MONTRÉAL

COLORATION DE GRAPHES ET ATTRIBUTION D'ACTIVITÉS DANS DES QUARTS DE TRAVAIL

MATHIEU BOUCHARD
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
OCTOBRE 2008

Your file Votre référence
ISBN: 978-0-494-46093-1
Our file Notre référence
ISBN: 978-0-494-46093-1

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

COLORATION DE GRAPHES ET ATTRIBUTION D'ACTIVITÉS DANS DES QUARTS DE TRAVAIL

présentée par : BOUCHARD Mathieu
en vue d'obtention du diplôme : Philosophiae Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. GAMACHE Michel, ing., Ph.D., président
M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche
M. HERTZ Alain, Doct. ès Sc., membre et codirecteur de recherche
M. HAHN Geňa, Doctorat d'état, membre
M. DE WERRA Dominique, Doct. ès Sc., membre externe

À Gaëlle, mon rayon de soleil

REMERCIEMENTS

Je remercie chaleureusement M. Guy Desaulniers, directeur de recherche, qui par ses conseils, ses idées, sa compétence et sa disponibilité a grandement facilité l'accomplissement de ce travail.

Je remercie aussi M. Alain Hertz, codirecteur de recherche, pour son aide et son enthousiasme ainsi que pour les idées de recherche très intéressantes qu'il m'a soumises.

Il va s'en dire qu'ils ont, tous deux, beaucoup enrichi le contenu de cette thèse et que ce fut un très grand plaisir d'étudier sous leur direction. Je les remercie aussi pour l'aide financière qu'ils m'ont fournie.

Merci à Mme Mirjana Čangalović, les moments passées à travailler avec elle furent des plus agréables.

Je voudrais en terminant remercier ma famille, mes parents pour toute l'aide qu'ils m'ont apportée et mes amours, Hélène et Gaëlle, pour tous ces moments heureux qui ont ensoleillé mes années d'études. Hélène, encore merci pour ton support et tes encouragements.

RÉSUMÉ

Les problèmes de fabrication d'horaires sont aussi nombreux que les contextes qui leur donnent naissance. Dans cette thèse, nous proposons des approches de résolution pour quelques uns de ces problèmes. Les méthodes proposées tentent d'exploiter autant que faire ce peut la structure sous-jacente du problème. Dans certains cas, cette structure est bien définie et le problème peut s'exprimer dans un contexte mathématique facilitant les liens avec des domaines bien établis de cette discipline. C'est le cas notamment des problèmes d'horaires dont la structure est similaire à celle de deux problèmes que nous étudierons dans cette thèse : le problème de minimisation de la déficience d'un graphe et le problème de coloration par intervalle qui sont étroitement liés aux problèmes très étudiés de coloration des nœuds d'un graphe et de coloration des arêtes d'un graphe. Dans le cas du problème d'attribution d'activités, un autre type de problème d'horaires étudié dans cette thèse, les structures qui ont été mises à profit sont celles de problèmes de flot associés à deux parties distinctes du problème complet.

Pour bien comprendre en quoi consiste le problème de minimisation de la déficience d'une coloration d'un graphe $G = (V, E)$, quelques définitions sont de mises. Une coloration des arêtes d'un graphe est une fonction qui assigne une couleur $c(e) \in I\!N$ à chaque arête de façon à ce que $c(e) \neq c(e')$ quand e et e' ont une extrémité en commun. Si l'on appelle $S_i(G, c)$ l'ensemble des couleurs appartenant aux arêtes incidentes à un nœud i et $D_i(G, c)$ le nombre minimum d'entiers à ajouter à $S_i(G, c)$ pour que celui-ci forme un intervalle (une suite consécutive de nombres entiers), alors la déficience $D(G, c)$ d'une coloration du graphe G est la somme $\sum_{i \in V} D_i(G, c)$ des déficiences de tous les nœuds. Le problème de minimisation de la déficience d'une coloration d'un graphe consiste à trouver une coloration c pour laquelle la déficience $D(G, c)$ du

graphe est minimale. Le premier article de cette thèse étudie ce problème NP-difficile. On présente d'abord une borne inférieure sur le nombre de couleurs nécessaires pour obtenir une coloration qui minimise $D(G, c)$. Nous proposons aussi un algorithme de recherche tabou permettant de résoudre le problème de minimisation de la déficience pour un graphe général et présentons des résultats numériques sur plusieurs types de graphes, pour certains de ceux-ci la déficience optimale est connue.

Le problème de coloration par intervalle, étudié dans les deuxième et troisième articles de cette thèse, s'applique à des graphes pondérés, c'est-à-dire qu'un poids ω_i, un entier strictement positif, est attribué à chaque nœud i du graphe. Une coloration par intervalle d'un graphe est une fonction qui attribue un intervalle de ω_i nombres entiers consécutifs (les couleurs) à chaque nœud i du graphe. Ces couleurs doivent être assignées de telle façon que deux nœuds adjacents n'ont aucune couleur en commun. Le problème de coloration par intervalle vise à trouver une coloration par intervalle utilisant le plus petit nombre de couleurs possible. Nous étudions d'abord la relation qui existe entre ce problème et un autre problème de coloration des nœuds d'un graphe, la coloration par bande. Dans le cas où un poids δ_{ij} est donné à chaque arête $\{i, j\}$ d'un graphe, une coloration par bande est une fonction c qui attribue un entier (la couleur) à chaque nœud de sorte que $|c(i) - c(j)| \geq \delta_{ij}$ pour toute arête $\{i, j\} \in E$. Le problème de coloration par bande vise à trouver une coloration par bande qui minimise la différence entre la plus grande et la plus petite couleur. Les résultats que nous avons obtenus montrent que le problème de coloration par intervalle peut être ramené à la résolution d'une série de problèmes de coloration par bande. Nous montrons aussi que le nombre de problèmes à résoudre est relativement petit. Les résultats expérimentaux obtenus montrent que la réduction peut aider à résoudre de grandes instances et à obtenir des bornes supérieures sur le nombre optimal de couleurs nécessaire pour une coloration par intervalle.

L'efficacité d'un algorithme pour résoudre le problème de coloration par intervalle peut être grandement améliorée si l'on évite de visiter des colorations équivalentes

lorsqu'on explore l'espace des solutions. Deux colorations par intervalle I_1 et I_2 d'un graphe $G = (V, E)$, utilisant k couleurs, sont dites équivalente si il existe une permutation π des entiers $1, \ldots, k$ telle que $\ell \in I_1(i)$ si et seulement si $\pi(\ell) \in I_2(i)$ pour tout nœud $i \in V$. Nous proposons et prouvons une condition nécessaire et suffisante pour que deux colorations utilisant k couleurs soient équivalentes. Nous montrons aussi comment un algorithme tabou simple pour le problème de coloration par intervalle peut être amélioré en évitant la visite de solutions équivalentes.

Le dernier sujet abordé dans cette thèse est celui de la résolution du problème d'attribution d'activités à des quarts de travail. Nous proposons d'abord une méthode de recherche tabou couplée à une technique de réduction afin de réduire la combinatoire du problème. Cette technique de réduction a beaucoup contribué à améliorer la méthode de recherche tabou. Suivant cette constatation, nous présentons quatre modèles de programmation linéaire en nombres entiers permettant de résoudre le problème d'attribution d'activités ; ces modèles sont inspirés des problèmes de flot à coût minimum utilisés dans la technique de réduction. Nous présentons ensuite les résultats de l'application d'une méthode de séparation et d'évaluation progressive commerciale, utilisant nos modèles, afin de résoudre des instances aléatoires inspirées de situations réelles. Les résultats montrent d'excellentes performances pour deux de ces modèles.

Pour chacun des problèmes considérés, cette thèse propose des méthodes nouvelles, fondées sur des considérations théoriques que nous avons établies et qui permettent d'évaluer leurs forces et leurs faiblesses. Les expérimentations numériques menées ont permis de démontrer l'efficacité de ces méthodes.

ABSTRACT

Scheduling problems are as diversified as the context in which they arise. In this thesis, we propose different approaches for solving some of these scheduling problems. Our methods aim at exploiting the underlying structure of the problem. In some cases, this structure is well known and the problem can be formulated in a sound mathematical context and links with other mathematical areas can be easily made. In particular, it is the case for scheduling problems having a structure similar to two problems studied in this thesis: the minimum deficiency problem and the interval coloring problem. These two problems are closely related to the well studied vertex coloring problem and edge coloring problem. In the case of the activity assignment problem, another problem studied in this thesis, we have exploited the network flow structure of two distinct sub-parts of the complete problem.

We first propose some definitions in order to get a better understanding of the minimum deficiency problem. An edge coloring of a graph $G = (V, E)$ is a function $c : E \rightarrow I\!N$ that assigns a color $c(e)$ to each edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' have a common endpoint. Denoting $S_i(G, c)$ the set of colors assigned to the edges incident to a vertex $i \in V$, and $D_i(G, c)$ the minimum number of integers which must be added to $S_i(G, c)$ to form an interval, the deficiency $D(G, c)$ of an edge coloring c is defined as the sum $\sum_{i \in V} D_i(G, c)$. The minimum deficiency problem is the problem of finding, for a given graph, an edge coloring with a minimum deficiency. The first part of this thesis studies this problem. We give new lower bounds on the minimum deficiency of an edge coloring and on the number of colors used in an edge coloring with minimum deficiency. We also propose a tabu search algorithm to solve the minimum deficiency problem and report experiments on various graph instances, some of them having a known optimal deficiency.

For the interval coloring problem, studied in the second and third papers of this thesis, we consider a graph $G = (V, E)$ with strictly positive integer weights ω_i on the vertices $i \in V$. An interval coloring of G is a function I that assigns an interval $I(i)$ of ω_i consecutive integers (called colors) to each vertex $i \in V$ such that $I(i) \cap I(j) = \emptyset$ for all edges $\{i, j\} \in E$. The interval coloring problem is to determine an interval coloring that uses as few colors as possible. The first thing we consider, concerning this problem, is how it is linked to another vertex coloring problem, the bandwidth coloring problem. Assuming that a strictly positive integer weight δ_{ij} is associated with each edge $\{i, j\} \in E$, a bandwidth coloring of G is a function c that assigns an integer (called a color) to each vertex $i \in V$ so that $|c(i) - c(j)| \geq \delta_{ij}$ for all edges $\{i, j\} \in E$. The bandwidth coloring problem is to determine a bandwidth coloring with minimum difference between the largest and the smallest colors used. We prove that an optimal solution of the interval coloring problem can be obtained by solving a series of bandwidth coloring problems. Computational experiments demonstrate that such a reduction can help to solve larger instances or to obtain better upper bounds on the optimal solution value of the interval coloring problem.

The efficiency of algorithms that solve the interval coloring problem can be increased by avoiding considering equivalent interval colorings. Two interval colorings I_1 and I_2, both using k colors, are said *equivalent* if there is a permutation π of the integers $1, \ldots, k$ such that $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$ for all vertices $i \in V$. In order to recognize equivalent colorings, we define and prove a necessary and sufficient condition for the equivalence of two interval colorings using k colors. We then show how a simple tabu search algorithm for the interval coloring problem can possibly be improved by forbidding the visit of equivalent solutions.

The last subject studied in this thesis is the activity assignment problem, in which we must assign different activities to pre-determined shifts. We first present a tabu search method combined with a reduction scheme in order to reduce the combinatorial complexity of the problem. The contribution of this reduction technic to the

performance of the tabu search is significant. Following this observation, we propose four different integer programming models, derived from the minimum flow problem used in the reduction technic, for solving the activity assignment problem. Then, we present the results of the computational experiments that we have conducted on randomly generated instances inspired by real-life situations. The instances were solved using a commercial branch-and-bound method using our four models. The results show great performance for two of these models.

For each studied problems, this thesis proposes new methods, based on theoretical considerations, that shed light on their strenghts and weaknesses. The computational experiments have shown the efficiency of the methods we developed.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

La nature des horaires de travail varie beaucoup selon le type de travail des employés pour qui l'on doit créer ces horaires. Dans certains cas, la tâche de fabriquer les horaires devient difficile et des outils mathématiques et informatiques sont nécessaires ou souhaitables lors de leur élabration. Cet état de fait a donné naissance, comme nous le verrons dans le prochain chapitre, a plusieurs travaux de recherche sur les problèmes concernant la planification des horaires de travail. Cette thèse a pour but de développer de nouvelles méthodes de résolution pour des problèmes de fabrication d'horaires ou pour des problèmes qui leur sont connexes. Dans un premier temps, nous étudions des problèmes de coloration d'arêtes et de noeuds avec certaines restrictions. Ces problèmes sont liés à la fabrication d'horaires dans la mesure où ils permettent de modéliser certains de ces problèmes qui présentent des contraintes demandant que des périodes soient attribuées consécutivement à certains événements (par exemple, les périodes d'un cours universitaire). Notons que tous les graphes considérés dans cette thèse sont finis. Ensuite, nous nous intéressons à un problème d'attribution des activités dans des quarts de travail. Ce chapitre présente tout d'abord ces deux problèmes pour ensuite présenter les pistes de recherche qui ont été suivies.

La création des horaires de travail peut devenir un vrai casse-tête lorsque la main-d'oeuvre est nombreuse et son utilisation complexe. Depuis les travaux de Edie (1954) et Dantzig (1954), les approches et les types de problèmes étudiés se sont beaucoup multipliés. Ernst *et al.* (2004a et b) ont classé ces problèmes en différentes catégories qui comptent notamment:

- les problèmes de construction de cycles (*days-off scheduling*);

- les problèmes de construction de quarts de travail (*shift scheduling*);
- les problèmes de construction simultanée de cycles et de quarts de travail (*tour scheduling*);
- les problèmes d'attribution d'activités à des quarts de travail;
- les problèmes d'affectation calendaire (*timetabling*);
- les problèmes de construction de rotations d'équipage (*crew scheduling*).

La construction de cycles consiste à établir les jours de travail et de repos pour un horizon temporel pré-établi et ce pour chaque employé. La construction de quarts de travail vise à déterminer les heures de travail (et dans certains cas, la nature de celui-ci) et les heures de pause pour un groupe d'employés donné. On peut aussi chercher à établir à la fois les cycles et les quarts de travail. Les problèmes d'attribution d'activités à des quarts de travail sont issus d'un contexte où la nature du travail (les activités) à l'intérieur d'un même quart de travail peut varier. Le problème consiste à établir la séquence des activités à effectuer pour chaque quart. Le problème d'affectation calendaire consiste à établir quand certains événements se tiendront (par exemple des cours ou des rencontres) considérant certaines restrictions et certaines possibilités de conflits. Le problème de construction de rotations d'équipage établit les heures de travail dans le cas spécifique où les employés se déplacent dans l'espace et où, dans la plupart des cas, il n'y a pas de possibilité de relève pour une tâche donnée. Plusieurs approches sont utilisées pour modéliser et résoudre ces problèmes, notons en particulier:

- intelligence artificielle (système expert, ensemble flou);
- programmation par contraintes;

- métaheuristiques (recuit simulé, recherche tabou, algorithme génétique, réseau neuronaux, procédure de recherche aléatoire adaptable, colonie de fourmis);
- programmation mathématique (modèle de recouvrement d'ensemble et de partition d'ensemble avec approche de génération de colonnes, formulation implicite équivalente au modèle de recouvrement avec une approche d'évaluation et séparation progressive).

Les problèmes d'attribution d'activités à des quarts considérés pour cette thèse sont de ceux qui se déroulent sur un horizon sans interruption et où les quarts de travail sont pré-déterminés. L'employé assigné à un quart possède des qualifications lui permettant d'effectuer un certain nombre d'activités. L'horizon est séparé en un ensemble de périodes disjointes de longueur égale. Pour chacune de ces périodes et pour chaque activité, une demande en employés nécessaires est donnée. On cherche à remplir les quarts de telle façon que la demande, pour chaque activité, soit couverte dans la mesure du possible. Pour une période donnée, il est possible d'avoir moins (resp. plus) de quarts couvrant une activité que ce qui est demandé pour cette activité. On dit alors que l'activité est en sous-couverture (resp. sur-couverture) pour la période. Pour éviter ces situations, on pénalise la sous-couverture et la sur-couverture. On définit, pour chaque activité, une durée minimale et maximale sur le nombre de périodes durant lesquelles un employé effectue cette activité sans interruption. Chaque quart comporte une ou plusieurs pauses qui sont, dans le cas qui nous intéresse, fixées et qui découpent le quart en pièces de travail. L'objectif prioritaire est de remplir les quarts de travail d'activités tout en minimisant la sous-couverture et la sur-couverture. Un objectif secondaire vise à diminuer le nombre de transitions d'une activité vers une autre à l'intérieur d'un même quart. Si une pause sépare deux activités différentes, le changement n'est pas considéré comme une transition. La pénalité pour les transitions est typiquement faible en comparaison à celle imposée pour la sous-couverture.

Les problèmes de coloration d'un graphe non orienté sont très répandus en théorie des graphes et, par conséquent, plusieurs résultats théoriques ont été établis en ce qui concerne leurs solutions et leurs modes de résolution. Le livre de Jensen et Toft (1995) est bonne référence portant sur les problèmes de coloration d'un graphe et sur plusieurs résultats théoriques d'importance les concernant. On peut séparer ces problèmes en deux familles selon que l'on cherche à donner des couleurs aux arêtes ou aux nœuds. Dans les problèmes de coloration de nœuds, deux noeuds adjacents ne peuvent avoir la même couleur, alors que dans les problèmes de coloration d'arêtes, toutes les arêtes incidentes à un même noeud doivent avoir des couleurs différentes. Chacune de ces familles de problèmes se déclinent en une multitude de variantes selon la façon dont on restreint la coloration des nœuds ou des arêtes. Une de ces variantes, la coloration consécutive des arêtes, que l'on a étudiée, vise à établir une correspondance entre les couleurs données aux arêtes et les nombres entiers de façon à ce que l'ensemble des nombres correspondant aux couleurs données aux arêtes incidentes à chaque nœud forme un intervalle de nombres entiers consécutifs.

Dans le cas où un graphe n'admet pas une telle coloration, un problème connexe, le problème de minimisation de la déficience du graphe, consiste à trouver une coloration dont l'écart (la déficience) avec une coloration séquentielle est minimal. Formellement, soit $G(V, E)$ un graphe où V est l'ensemble des nœuds et E l'ensemble des arêtes. Une coloration c des arêtes du graphe G est une fonction $c : E \rightarrow I\!N$ telle que si les arêtes $a, a' \in E$ sont incidentes à un même nœud alors $c(a) \neq c(a')$. On définit la déficience d'un nœud $i \in V$, dénoté par $D_i(G, c)$, comme étant $max(i) - min(i) - \Delta(i) + 1$ où $max(i)$ (resp. $min(i)$) est le nombre le plus élevé (resp. moins élevé) affecté aux arêtes incidentes à i ($max(i) = \max_{e \in E_i} c(e)$, $min(i) = \min_{e \in E_i} c(e)$), $\Delta(i)$ est le nombre d'arêtes incidentes à i et E_i est l'ensemble des arêtes incidentes à i. La déficience $D(G, c)$ d'une coloration c du graphe G est la somme des déficiences de ces nœuds ($D(G, c) = \sum_{i \in V} D_i(G, c)$). Le problème de minimisation de la déficience d'une coloration du graphe G est celui de déterminer une

coloration c de G qui minimise $D(G, c)$. Le problème visant à obtenir une coloration c de G pour laquelle $D(G, c) = 0$ est nommé le problème de coloration consécutive des arêtes d'un graphe.

Nous présentons à la figure 1 deux petits exemples de coloration qui illustrent les deux problèmes de coloration auxquels nous nous intéressons. Le graphe représenté à la figure 1 a) est une coloration consécutive, c'est-à-dire les couleurs des arêtes incidentes à chaque nœud forment un intervalle de nombres entiers. Par exemple, les arêtes incidentes au nœud g forment l'intervalle $[2, 4]$. Il est à noter que le nombre de couleurs minimum nécessaires pour colorier consécutivement ce graphe est supérieur à l'indice chromatique du graphe (le plus petit nombre de couleurs d'une coloration sans conflit) qui est de 3. Le graphe représenté à la figure 1 b) est un graphe qui ne peut être colorié consécutivement: il y aura toujours au moins un nœud dont la déficience est non nulle. Dans ce cas, seul le nœud i a une déficience non nulle: celle-ci est de $3 - 1 - \Delta(i) + 1 = 1$. La déficience de G, $D(G)$, est donc égale à 1.

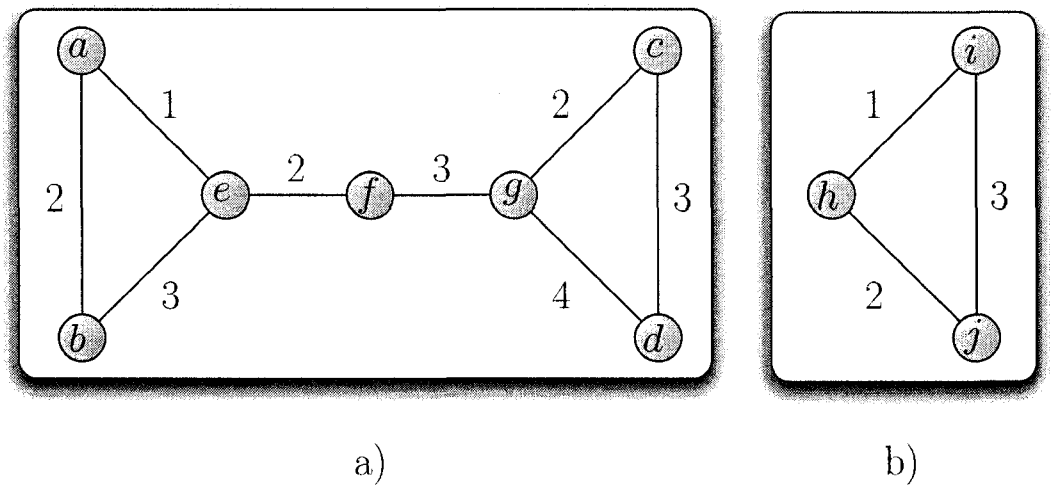


Figure 1: Deux exemples de coloration des arêtes d'un graphe.

On retrouve une contrainte similaire dans une généralisation du problème de col-

oration des nœuds d'un graphe dans laquelle on peut donner plus d'une couleur aux nœuds d'un graphe. Dans le problème de coloration par intervalle, on cherche une coloration utilisant le plus petit nombre de couleurs (entiers positifs) tel qu'à chaque noœud est associé un nombre donné d'entiers positifs consécutifs. Encore là, deux nœuds adjacents ne doivent pas se voir attribuer la même couleur. Plus précisément, soit $G(V, E)$ un graphe où V est l'ensemble des nœuds et E l'ensemble des arêtes. Chaque nœud i a un poids w_i qui détermine le nombre de couleurs devant lui être attribuées. Une coloration c des nœuds du graphe G est une fonction $c : V \rightarrow I\!N$ telle que si les nœuds $i, j \in V$ sont adjacents alors soit $c(i)+w_i \leq c(j)$ soit $c(j)+w_j \leq c(i)$. Les entiers consécutifs (couleurs) assignées à i sont $\{c(i), \ldots, c(i)+w_i-1\}$. On cherche une coloration qui minimise $\max_{i \in V}(c(i) + w_i - 1)$. Ce problème sera aussi étudié. La figure 2 présente une coloration optimale pour le problème de coloration par intervalle où le poids des nœuds a, c, d est de 2 et le poids du nœud b est de 3, dans ce cas on a $c(a) = 1$, $c(b) = 3$, $c(c) = 6$ et $c(d) = 1$.

Plusieurs problèmes d'affectation calendaire peuvent être modélisés comme des problèmes de coloration. Par exemple, considérons l'attribution, sur l'horizon le plus court possible, des heures de début de différentes séances de cours, où certains cours ne peuvent se dérouler à la même heure et où les cours ont tous la même durée. Ce problème peut directement se modéliser comme un problème de coloration de nœuds où les nœuds du graphe sont les séances de cours et où une arête entre deux nœuds illustrent le fait que deux séances de cours ne peuvent être attribuées à la même heure. Si l'horizon temporel est divisé en périodes de longueur égale à la durée d'un cours, la couleur donnée à un nœud détermine alors la période durant laquelle se donne le cours correspondant. Le problème de coloration par intervalle permet d'étendre cette modélisation à des cours n'ayant pas tous la même longueur.

Dans un contexte similaire, un problème de coloration d'arêtes permet de modéliser le problème visant à attribuer des périodes d'enseignement, de même longueur, pour un

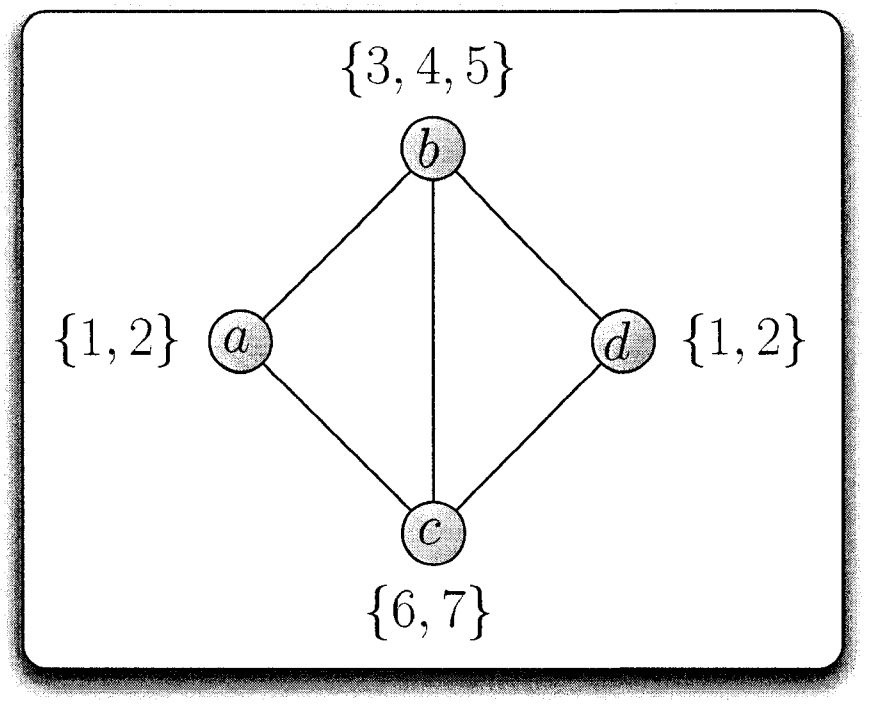


Figure 2: Un exemple de coloration par intervalle.

ensemble de professeurs qui enseignent à un ensemble de groupes. Dans ce modèle, on définit un graphe biparti, où un ensemble de nœuds correspond aux professeurs, un autre ensemble de nœuds correspond aux groupes et une arête indique que l'on doit assigner une période d'enseignement pour le professeur et le groupe correspondant aux extrémités de l'arête. Si l'horizon temporel est divisé en périodes de longueur égale à la durée d'une période d'enseignement, la couleur de cette arête dans la solution correspond à la période à laquelle doit se tenir la rencontre.

C'est ce lien entre les problèmes de coloration de graphes et différents problèmes de planification d'horaires qui nous a amenés, dans cette thèse, à nous intéresser aux résultats théoriques, concernant ces problèmes, qui seraient susceptibles d'aider à leur résolution. Suite à cette recherche, nous proposons de nouveaux algorithmes,

basés sur la méthode tabou (voir Glover et Laguna 1997), permettant de résoudre des instances du problème de minimisation de la déficience d'un graphe et du problème de coloration par intervalle pour des graphes quelconques.

Cette recherche se termine par la présentation d'une nouvelle piste de résolution, cette fois pour un problème précis de planification d'horaires, soit le problème d'attribution des activités dans des quarts de travail présenté précédemment. Ayant eu du succès avec la méthode tabou pour la résolution des deux problèmes de coloration de graphe étudiés, c'est cette méthode que nous avons d'abord essayée pour ce dernier problème. Mais c'est finalement une formulation de celui-ci comme un programme linéaire en nombres entiers et sa résolution à l'aide d'outils commerciaux qui ont donné les meilleurs résultats.

Le premier chapitre de cette thèse présente une revue de la littérature. Celle-ci couvre d'abord le problème d'attribution des activités à des quarts de travail ainsi que des problèmes de fabrication d'horaires similaires. Ensuite, nous étudions les travaux portant sur la coloration séquentielle des arêtes d'un graphe et sur la coloration par intervalle. Le deuxième chapitre expose l'organisation de la thèse. On y décrit sommairement les recherches menées dans cette thèse dans un contexte unifié. Le troisième chapitre présente les résultats théoriques et les méthodes que nous avons développées pour le problème de minimisation de la déficience d'un graphe. Dans le quatrième chapitre, nous présentons une méthode heuristique qui résout le problème de coloration par intervalle en le transformant en un problème de coloration similaire. Le cinquième chapitre propose une condition nécessaire et suffisante permettant de déterminer si deux colorations par intervalle sont équivalentes. On y montre aussi comment tirer profit de cette condition, dans une méthode de recherche tabou, afin de réduire l'espace des solutions. Le dernier chapitre propose deux méthodes, une heuristique, l'autre exacte, pour la résolution du problème d'attribution des activités à des quarts de travail. Finalement, la conclusion met en lumière les contributions

apportées par cette thèse et propose des pistes de réflexion pour des recherches futures.

CHAPTER 1 : REVUE DE LITTÉRATURE

La revue de littérature présentée dans ce chapitre couvre les deux types de problèmes de coloration étudiés dans cette thèse et les problèmes de construction d'horaires. Une attention particulière est mise sur les travaux plus directement liés à nos recherches. La présentation de la littérature est séparée de la façon suivante:

- construction de cycles;
- construction de quarts de travail;
- construction simultanée de cycles et de quarts de travail;
- affectation calendaire;
- coloration consécutive des arêtes d'un graphe;
- coloration par intervalle.

En ce qui concerne les travaux portant sur les problèmes de construction d'horaires, notons que Ernst *et al.* (2004a et b) ont proposé une revue de littérature exhaustive des problèmes de création d'horaires et établi une classification à la fois complète et précise de ces problèmes. Le lecteur y trouvera plusieurs articles qui viennent compléter cette revue de littérature. Dans cette étude, les auteurs ont répertoriés et classifiés 1114 articles, ce qui donne une bonne idée de la pluralité du domaine.

Considérant l'étendue de ce secteur de recherche, nous avons choisi de restreindre la revue de littérature sur la construction d'horaires aux problèmes de construction de quarts de travail, de construction simultanée de cycles et de quarts de travail, d'attribution des activités dans des quarts de travail et d'affectation calendaire.

1.1 Construction de cycles

La construction de cycles a pour but d'établir les jours de congé et de repos des employés pour un horizon de planification donné. Plusieurs fonctions objectifs peuvent êtres considérées, l'objectif visant à minimiser le nombre d'employés est celui qui revient le plus souvent. On peut aussi trouver différents niveaux de souplesse, qui consistent généralement en un choix entre différents patrons de jour de travail. Quoiqu'un degré de souplesse plus élevé peut considérablement complexifier le problème, Mabert et Watts (1982) ont montré qu'il a souvent un impact positif sur l'utilisation du personnel. Deux approches parmi les méthodes proposées pour résoudre le problème de construction des cycles ont retenu notre attention. Burns et Carter (1985) proposent une méthode constructive basée sur une borne inférieure sur le nombre minimum d'employés nécessaire pour respecter les contraintes du problème. Ils ont construit un algorithme itératif qui permet d'obtenir un horaire réalisable et qui peut être utilisé dans le cas où l'horizon n'est pas continu (i.e. il existe au moins une période par jour où il n'y a pas de demande). Emmons et Burns (1991) présentent un algorithme glouton exact fonctionnant sur un horizon continu et ayant la particularité d'inclure différents niveaux de qualification. Un horizon est continu lorsqu'il n'y a pas de période où la demande est nulle.

1.2 Construction de quarts de travail

Edie (1954) est à l'origine des premiers travaux sur la construction de quarts de travail. Les problèmes étudiés consistaient à construire pour environ 250 policiers des quarts de travail afin de les affecter à un certain nombre de postes de péage. Des outils statistiques permettaient de déterminer le nombre de policiers nécessaires à chaque période de la journée. La technique retenue pour résoudre ces problèmes en

était une d'essais et erreurs qui était fastidieuse et qui ne permettait pas de savoir si une solution donnée était optimale.

Dantzig (1954) a proposé une première formulation basée sur un modèle généralisé de recouvrement d'ensemble dans lequel chaque quart (un pour chaque combinaison de temps de début, temps de fin et de placement de pauses) est représenté explicitement par une variable de décision. Le nombre de quarts à énumérer avec ce modèle croît extrêmement rapidement lorsque la souplesse est augmentée: le problème de recouvrement devient alors très difficile à résoudre. Segal (1974), Keith (1979) et Henderson et Berry (1976) ont étudié l'utilisation d'approches heuristiques pour contourner cette difficulté.

Moondra (1976) a utilisé une modélisation où la flexibilité sur la longueur des quarts et les différents temps de début des quarts est représentée implicitement. Il a considéré des quarts à temps plein de durée fixe contenant une pause pouvant débuter à l'intérieur d'une fenêtre de temps de deux périodes et des quarts à temps partiel de longueur flexible ne contenant pas de pause. La flexibilité sur la durée des quarts à temps-partiel est modélisée à l'aide de variables définissant les temps de début et de fin de quart et un ensemble de contraintes qui force le respect des durées maximum et minimum de ces quarts. Afin de tenir compte de la flexibilité lors du placement des pauses, la moitié des employés à temps-plein sont forcés de prendre leur pause lors de la première période de la fenêtre de pause et les autres à la seconde période.

Les auteurs précédents sont des pionniers dans la résolution du problème de construction de quarts de travail. Leurs travaux ont inspiré fortement les approches plus récentes. Bechtold et Jacobs (1990) ont proposé une approche différente qui consiste à réduire la taille du modèle en représentant implicitement le placement des pauses. L'algorithme proposé fonctionne sur un horizon non continu où une pause unique non décomposable peut débuter dans une fenêtre de temps donnée. La durée

de cette pause est fixe. Dans cette formulation, chaque quart (un pour chaque combinaison de temps de début et de durée de quart) est représenté par une variable indiquant le nombre de quarts de ce type devant être générés. Pour chaque période de l'horizon d'optimisation comprise entre la première période et la dernière période où une pause peut débuter, le modèle associe une variable indiquant le nombre de pauses commençant à cette période. Les variables de quart et de pause sont liées par des contraintes de type *en avant* et *en arrière* demandant qu'il y ait un nombre suffisant de pauses avant et après un ensemble de temps précis. Lorsque la flexibilité sur les pauses est assez grande, le modèle de Bechtold et Jacobs permet de diminuer le nombre de variables requises par rapport au modèle généralisé de recouvrement d'ensemble présenté par Dantzig (1954). Le modèle mathématique est de type programmation linéaire en nombres entiers (PLNE) et est résolu à l'aide d'un algorithme de recherche de solutions entières avec plans coupants. Leur algorithme permet de trouver, en moins de 20 minutes, la solution optimale pour des instances ayant entre 5 et 45 employés, un horizon de 12 heures (la période de discrétisation est de 30 minutes) et 970 types de quarts.

Thompson (1995) a présenté un modèle hybride reprenant la représentation implicite des quarts du modèle de Moondra (1976) et qui s'inspire du modèle de Bechtold et Jacobs (1990) pour représenter implicitement le placement des pauses dans le contexte d'une planification sur un horizon continu. Chaque quart reçoit au plus une pause sujette à certaines restrictions concernant les durées de travail survenant avant et après celle-ci. Les quarts ayant les mêmes coûts par période de travail, les mêmes possibilités de durée, les mêmes durées de pause et les mêmes restrictions sur le placement de la pause sont regroupés dans un même type de quart. Pour chaque type de quart, des variables sont définies pour représenter le temps de début et de fin du quart et le temps de début de la pause. Un ensemble de contraintes permet de lier ces différentes variables afin d'assurer la conformité des quarts. Le modèle

mathématique présenté par l'auteur est de type PLNE et est résolu par un algorithme exact de séparation et évaluation progressive. L'auteur a obtenu des solutions en 7 minutes sur les mêmes problèmes que Bechtold et Jacobs (1990).

Aykin (1996) a présenté une approche permettant de résoudre le problème de construction de quarts. Le modèle qu'il propose permet de planifier plusieurs types de pauses par quart, chacune pouvant débuter dans une fenêtre de temps pré-déterminée. Le placement des pauses est représenté en introduisant une variable pour chaque quart et chaque période où peut commencer une pause. Un quart est défini comme la combinaison d'un temps de début, une durée, les types de pauses qu'il contient et les fenêtres qui leur sont associées. Contrairement à la modélisation présentée par Bechtold et Jacobs (1990), ce modèle permet plusieurs types de pauses et le chevauchement extraordinaire. Il y a chevauchement extraordinaire lorsqu'il existe deux quarts pour lesquels une fenêtre de pause de l'un commence strictement avant une fenêtre de pause de l'autre quart et se termine strictement après la même fenêtre de pause de l'autre quart. Le pairage entre les quarts et les pauses pouvant leur être associées est obtenu par un ensemble de contraintes d'égalité, une pour chaque type de pause et chaque quart. Aykin (1998) propose de résoudre la formulation de Aykin (1996) à l'aide d'un algorithme de séparation et évaluation progressive combiné à des plans coupants. Aykin (2000) a montré que, pour une certaine famille de problèmes, sa formulation permet d'obtenir plus rapidement des solutions optimales que la formulation de Bechtold et Jacobs (1990).

Bechtold et Jacobs (1996) ont prouvé que la formulation implicite qu'ils ont élaborée et la formulation de recouvrement sont équivalentes en nombres entiers sous certaines conditions. Çezik et Günlük (2004) et Rekik *et al.* (2004) ont indépendamment prouvé une version plus forte de cette équivalence en ne gardant qu'une seule condition concernant les pauses. Addou (2005) a proposé une extension du modèle de Bechtold et Jacobs (1990) pour lequel l'équivalence en nombres entiers avec la formulation de recouvrement ne demande pas de condition particulière.

En plus des articles de synthèse de Ernst *et al.* (2004a et b), le mémoire d'Addou (2005) constitue une autre bonne référence sur les travaux concernant la construction de quarts de travail.

1.3 Construction simultanée de cycles et de quarts de travail

La construction simultanée des cycles et des quarts de travail consiste à résoudre à la fois le problème de construction des quarts et de construction des cycles. Les possibilités de flexibilité présentes dans ces deux problèmes peuvent aussi se retrouver dans le problème de construction des cycles et des quarts.

Burns et Koop (1987) ont proposé un algorithme itératif permettant de générer un horaire utilisant le minimum d'employés sur un horizon continu. Cet algorithme combine des horaires de petite taille qui sont prédéterminés. L'approche qu'ils ont présentée ne permet pas la planification des pauses et aucune flexibilité n'est possible quant aux heures de début et aux durées des quarts. Leur algorithme est simple mais son champ d'application est limité.

Easton et Rossin (1991) ont présenté une heuristique utilisant une approche par génération de colonnes. La méthode, qui fonctionne sur un horizon non continu, ne permet pas la planification des pauses et ne considère pas les patrons de quarts (un patron de quarts est une séquence, de longueur variable, d'alternatives de types de quarts que les employés désirent travailler). Elle permet, par contre, une flexibilité sur les heures de début de quart. Une heuristique réduit la taille du problème en choisissant un sous-ensemble d'horaires de travail parmi tous les horaires réalisables. Les résultats obtenus sont supérieurs à ceux obtenus avec d'autres méthodes utilisant un sous-ensemble prédéterminé d'horaires possibles.

Jarrah *et al.* (1994) ont présenté un algorithme heuristique dans le contexte d'un horizon non continu. Le modèle ne considère pas les patrons de quarts. Cependant, il prend en charge la planification d'une pause à l'intérieur d'une fenêtre de temps, considère divers types de quarts et permet une flexibilité sur les heures de début et les durées des quarts. L'algorithme présenté permet d'obtenir des solutions de bonne qualité en des temps raisonnables.

Brusco et Jacobs (1998) ont étudié le cas où il y a des contraintes qui imposent un délai minimal entre les heures successives de début de quarts. Les auteurs ont divisé le problème en deux parties: la détermination des heures de début des quarts et l'affectation des cycles aux employés. Quoique l'algorithme présenté soit assez rapide, il ne trouve pas de solution pour certaines instances du problème considéré.

Haase (1999) a proposé un algorithme exact pour un problème sur un horizon continu. La méthode permet de prendre en considération la planification de plusieurs pauses et l'utilisation de fenêtres de temps pour les pauses ainsi que la flexibilité sur la durée et l'heure de début des quarts. L'auteur propose un modèle de recouvrement généralisé qui se résout par une méthode de génération de colonnes couplée à un algorithme exact de séparation et évaluation progressive.

Topaloglu *et al.* (2002) présentent un modèle implicite de construction simultanée des cycles et des quarts en utilisant une combinaison du modèle d'Aykin avec celui proposé par Bailey (1985). Ils ont travaillé sur des journées de travail de moins de 24 heures pendant 7 jours dont deux jours de repos par semaine.

Bard *et al.* (2002) utilisent le modèle de Bechtold et Jacobs en y ajoutant quelques contraintes introduites par Burns (1985) afin de donner aux employés deux jours de repos consécutifs. Comme Topaloglu *et al.* (2002), ils considèrent des journées de travail de 24 heures sur 7 jours dont deux jours de repos.

Rekik *et al.* (2004) ont présenté une approche généralisant l'utilisation des contraintes en avant et en arrière introduite par Bechtold et Jacobs (1990) permettant ainsi de l'appliquer au problème de construction des cycles et des quarts.

1.4 Attribution des activités dans des quarts de travail

Omari (2002) a proposé un algorithme heuristique permettant de résoudre le problème d'affectation d'activités à l'intérieur des quarts sur un horizon continu pour les contrôleurs aériens. Il s'agit ici d'un contexte multi-activités. Partant de quarts de travail construits et déjà affectés aux employés, la méthode présentée par l'auteur remplit les quarts par l'affectation d'activités. Le modèle tient compte des compétences des employés lors de l'affectation des activités. L'objectif du problème considéré vise à minimiser la somme des sous-couvertures des demandes pour chaque activité et chaque période de l'horizon de planification. Les problèmes doivent respecter de nombreuses règles issues de la convention collective des employés. La méthode choisie suit l'approche de recouvrement d'ensemble proposée par Dantzig, à la différence qu'une technique de résolution basée sur une méthode de génération de colonnes combinée à un algorithme heuristique de séparation et évaluation progressive est utilisée pour contourner la difficulté associée au grand nombre de variables. En effet, dans ce genre de modèle, il y a au moins une variable par quart possible et la dimension multi-activités du problème induit un grand nombre de quarts possibles. Pour chaque quart de travail, un problème de plus court chemin est utilisé pour générer une nouvelle colonne qui correspond à une variable, représentant un quart de travail, de plus petit coût réduit. Certains arcs de ce graphe modélisent une transition et ont un coût positif; d'autres modélisent l'assignation d'une activité à une ou plusieurs périodes du quart. Une décomposition de Dantzig-Wolfe permet

d'attribuer un coût à ces arêtes afin que la résolution du problème de plus court chemin revienne à trouver la colonne de plus petit coût réduit. Les jeux de données utilisés comprennent un horizon d'un mois, 85 employés, une vingtaine d'activités ainsi qu'une discrétisation de l'horizon de planification en période de 15 minutes. Les résultats obtenus par l'auteur présente une diminution de la somme des sous-couvertures par rapport aux solutions obtenues à l'aide des méthodes précédemment utilisées pour ce problème spécifique.

La fabrication d'horaires avec quarts de travail consiste à combiner le problème de fabrication de quarts et le problème d'affectation des activités à l'intérieur des quarts. Vatri (2001) a présenté une extension des travaux de Omari (2002) permettant la résolution de ce problème mixte. En effet, il reprend le modèle de Omari (2002) pour l'affectation des activités et y ajoute des composantes de construction de quarts et d'affectation des quarts aux employés. Les problèmes étudiés sont les mêmes que ceux de Omari (2002) et se situent donc dans un contexte multi-activités sur un horizon continu. Le modèle présenté permet une flexibilité sur la durée et le début des quarts. De plus, des contraintes permettent de respecter des règles régissant le nombre de quarts possédant certaines caractéristiques à générer. Les jours de travail et les patrons de quarts des employés sont pré-assignés. Le modèle présenté ne permet toutefois pas la planification des pauses à l'intérieur des quarts de travail. L'auteur utilise une méthode heuristique de séparation et évaluation progressive intégrant une approche de génération de colonnes combinée à une stratégie de décomposition temporelle. Les résultats présentés montrent que la méthode permet d'obtenir des solutions, pour la fabrication d'horaires avec quarts de travail, légèrement meilleures que celles obtenues avec les méthodes précédentes utilisant des quarts pré-établis.

Bouchard (2004) a étendu les travaux de Vatri (2001) afin de permettre une flexibilité lors du placement des pauses à l'intérieur des quarts. Il a aussi démontré que l'intégration de la construction des quarts de travail avec pauses, de l'assignation

des quarts aux employés et l'attribution des activités permet d'obtenir des solutions de meilleure qualité. La technique de résolution utilisée intègre une méthode de génération de colonnes combinée à une stratégie de décomposition temporelle.

Notons que certains de ces travaux, portant sur l'attribution des activités dans des quarts de travail, sont présentés de façon plus approfondie au chapitre 6.

1.5 Affectation calendaire

Nous présentons ici les travaux sur ce sujet qui ont un lien direct avec les problèmes de coloration qui nous intéressent.

Punter (1976) a formulé et résolu à l'aide d'un modèle de coloration par intervalle, un problème d'horaires en milieu scolaire avec des leçons couvrant plusieurs périodes et ne pouvant être interrompues. Le problème classique de coloration des nœuds ne permettait pas de modéliser le cas des leçons couvrant plusieurs périodes et ne pouvant être interrompues, c'est pourquoi Punter a introduit l'idée d'attribuer plusieurs couleurs à chaque nœud afin de contourner cette limitation.

Bartholdi *et al.* (1990) se sont penchés sur un problème consistant à affecter des rencontres entre des employeurs potentiels (entre 25 et 50) et des étudiants à la recherche d'emplois (100 et 200). Les disponibilités offertes pour les rencontres sont des tranches de temps prédéterminées que l'on peut énumérer de 1 à n (où n est environ 25). Puisque ni les employeurs, ni les étudiants n'ont plus de n rencontres de planifiées, le problème correspond à colorier les arêtes d'un graphe biparti avec n couleurs où deux arêtes adjacentes n'ont pas la même couleur. Puisque le degré de chaque nœud correspond au nombre de rencontres de l'intervenant auquel le nœud correspond et que celui-ci est inférieur ou égal à n, alors n couleurs suffisent pour colorier les arêtes du graphe (voir Gibbons, 1985). Les auteurs soulignent que plusieurs

participants, autant des employeurs que des étudiants, ont fait la demande que les rencontres soient affectées consécutivement (sans temps mort entre deux rencontres). Les auteurs proposent un algorithme glouton simple qui a un temps d'exécution de $O(|E|^2)$ dans le pire cas, où E est l'ensemble des arêtes du graphe, tout en ayant des propriétés fort utiles dans son contexte d'application. Quoique l'algorithme proposé ne cherche pas à obtenir une coloration qui minimise la déficience (le nombre de temps morts), les auteurs démontrent que, d'un certain point de vue, la déficience résultant de la coloration obtenue est petite. Notons que Gabow (1976) propose un algorithme ayant un temps d'exécution de $O(n^{1/2}|E| \log n+n)$ dans le pire cas et Gabow et Kariv (1982) étendent cette méthode pour obtenir un temps d'exécution dans le pire cas de $O(|E|(\log n)^2)$ et de $O(n^2 \log n)$. L'avantage de leur méthode réside dans sa simplicité et dans les avantages qu'elle procure dans son contexte d'application, notamment en ce qui a trait à la réduction des temps morts.

de Werra (1997) a étudié des problèmes d'affectation calendaire qui peuvent se traduire comme des problèmes de coloration des nœuds d'un graphe où certaines restrictions s'appliquent: la couleur de chaque nœud n du graphe doit appartenir à un ensemble restreint $\varphi(n)$ de couleurs et le nombre total de couleurs pouvant être utilisé est limité. Il utilise une formulation qu'il reprend dans de Werra (1999). Il étudie en particulier les cas où la matrice de contraintes résultante est parfaite, équilibrée ou totalement unimodulaire. De plus, l'auteur présente une revue des différents résultats et variantes présents dans la littérature.

Asratian et de Werra (2002) étudient le problème d'affectation calendaire consistant à affecter des classes à des professeurs dans le cas particulier où, en plus d'affecter des professeurs à des classes individuelles, certains professeurs doivent donner des cours à des classes multiples (l'ensemble des classes est divisé en groupes de classes nommées classes multiples). Ils établissent la $\mathcal{NP}$-complétude du problème de décision associé à ce problème et fournissent, considérant certaines restrictions, un algorithme permettant d'obtenir une solution dont la longueur est au plus $\frac{7}{6}$ de la durée optimale. de

Werra *et al.* (2002) présentent des résultats de complexité pour des cas particuliers de ce problème. Ils établissent que, lorsqu'il y a au moins un professeur qui doit donner des cours à au moins 3 classes multiples, le problème devient $\mathcal{NP}$-complet. Dans le cas où l'ensemble des classes est divisé en 2 classes multiples, les auteurs présentent un algorithme polynomial permettant de trouver un horaire ayant une durée d'au plus t périodes.

1.6 Coloration par intervalle

Comme nous l'avons vu, le problème de coloration par intervalle a été introduit par Punter (1976) pour modéliser un problème de fabrication d'horaires en milieu scolaire avec plus de souplesse. Clementson et Elphick (1983) ont aussi modélisé ce genre de problèmes comme un problème de coloration de graphe, notamment, comme un problème de coloration par intervalle et un problème de coloration des arêtes par intervalle (i.e. une coloration des arêtes d'un graphe où certaines arêtes doivent se voir attribuer plus d'une couleur). Ils présentent aussi quelques algorithmes approximatifs pour résoudre ces problèmes.

de Werra et Hertz (1988) ont établi des bornes supérieures sur le nombre minimum de couleurs à utiliser pour une coloration par intervalle. Ils ont aussi étudié le cas d'une sous-classe des graphes parfaits. Kubale (1989) ajoute quelques restrictions aux colorations par intervalle en interdisant, pour chaque nœud, un certain nombre de couleurs et donne des bornes inférieures et supérieures sur le nombre de couleurs requises. Une analyse de complexité est menée pour ce problème étendu.

Čangalović et Schreuder (1991) proposent un algorithme exact pour la coloration par intervalle. L'algorithme procède par énumération selon un principe similaire à la technique de séparation et évaluation progressive. Des expériences numériques

ont été menées sur des graphes aléatoires et des instances de planification d'horaires en milieu scolaire. L'efficacité de l'algorithme dépend en grande partie de l'ordre dans lequel on classe les nœuds initialement. Cet algorithme servira de point de comparaison à une des méthodes présentées dans cette thèse. Les auteurs font aussi l'observation que, si tous les poids sont égaux, le problème est équivalent au problème classique de coloration des nœuds.

Il existe dans la littérature des problèmes de coloration similaires à la coloration par intervalle. Le problème de coloration par bande consiste à colorier les nœuds d'un graphe (une couleur par nœud) ayant un poids sur chaque arête. Chaque couleur correspond à un nombre entier et la différence, en valeur absolue, des couleurs attribuées à des nœuds adjacents, doit être plus grande que le poids de l'arête qui les unit. On cherche la coloration qui, si les couleurs sont strictement positives, utilise les couleurs les plus petites possibles. Notons que la coloration classique des nœuds est un cas particulier de ce problème où tous les poids valent 1. Pour un problème de coloration par intervalle donné, il est possible d'attribuer des poids aux arêtes du graphe de telle sorte qu'une coloration par bande valide peut toujours être traduite en une coloration par intervalle valide. Cependant, l'objectif diffère quelque peu et les problèmes ne sont pas tout à fait équivalents. Prestwich (2002) a développé une méthode exacte pour ce problème qui peut aussi être utilisée de façon heuristique. Un des sujets de cette thèse est de réduire le problème de coloration par intervalle à une série de problèmes de coloration par bande. Lors de notre expérimentation numérique, nous utiliserons l'algorithme de Prestwich pour obtenir des colorations par bande utilisant les plus petites couleurs.

Un autre problème similaire est le problème de coloration p-circulaire sur des nœuds pondérés introduit par Deuber et Zhu (1996). Dans ce cas aussi, le nombre de couleurs est dicté par le poids d'un nœud et on cherche à obtenir pour chaque nœud des couleurs consécutives et ce en utilisant seulement les couleurs $\{0, \ldots, p-1\}$. La

différence est que l'on considère que 0 suit $p-1$. Ce problème n'est pas non plus équivalent à la coloration par intervalle.

1.7 Coloration consécutive des arêtes d'un graphe

Les problèmes de coloration consécutive considérés ici sont ceux qui concernent la coloration des arêtes d'un graphe. Le problème de coloration consécutive des arêtes d'un graphe G est équivalent aux problèmes de coloration des nœuds du graphe représentatif des arêtes (*line graph*) de G avec la restriction que les couleurs attribuées aux nœuds d'une clique du graphe de ligne, correspondant à des arcs incidents à un même nœud dans le graphe G, forment un intervalle. Ce problème de coloration de nœuds est différent de ceux présentés précédemment. Nous n'avons pas trouvé dans la littérature d'articles traitant de cette formulation équivalente.

Un petit groupe de gens ont étudié les problèmes de coloration consécutive. Plusieurs de ces travaux cherchent à établir certaines caractéristiques des colorations consécutives. La plupart des autres recherches concernent des variantes du problème permettant de modéliser un problème spécifique. Concernant les problèmes de minimisation de la déficience d'un graphe, il semble que Giaro (1999) soit le seul à s'y être intéressé. À notre connaissance, personne n'a proposé d'approche de résolution générale et efficace pour ces deux problèmes.

Cette partie de la revue de littérature est divisée en deux parties: la première couvre les différentes connaissances théoriques établies en ce qui concerne les problèmes de coloration consécutive; la deuxième partie s'attarde sur les applications courantes des problèmes de coloration consécutive, soit les problèmes d'ordonnancement. Rappelons que l'on a couvert dans la revue de littérature concernant les problèmes d'horaires quelques applications des problèmes de coloration consécutive et de coloration par intervalle pour résoudre des problèmes d'affectation calendaire.

1.7.1 Résultats théoriques

L'article d'Asratian et Kamalian (1994) constitue un bon point de départ pour considérer l'étendue des connaissances sur le problème de coloration consécutive d'arêtes. En particulier, ils ont recensé et synthétisé plusieurs des travaux publiés en russe par Asratian qui fut l'un des premiers à étudier ces problèmes dans les années 1980. On y retrouve aussi les résultats suivants, qui sont fondamentaux aux problèmes de coloration d'arêtes:

- Soit un graphe $G = (N; E)$ et $\Delta(G) = \max_{n \in N} \Delta(n)$ où $\Delta(n)$ est le nombre d'arêtes incidentes au nœud n. Vizing (1965) a établi que $\chi'(G)$, le nombre minimum de couleurs nécessaires pour colorier les arêtes d'un graphe, est $\Delta(G)$ (alors G est dit de classe 1) ou $\Delta(G) + 1$ (G est alors dit de classe 2).
- Holyer (1981) a montré que le problème consistant à déterminer si G est de classe 1 ou de classe 2 est $\mathcal{NP}$-complet.

De plus, Asratian et Kamalian (1994) établissent les résultats suivants:

- Si la déficience de G est 0, alors G est de classe 1.
- Si la déficience de G est 0, alors une coloration consécutive est possible avec au plus $2n(G) - 1$ couleurs, où $n(G)$ est le nombre de nœuds. De plus, si G n'a pas de triangle (boucle de longueur 3), alors ce nombre devient $n(G) - 1$.
- Si la déficience de G est 0 et G est biparti, alors une coloration consécutive est possible avec au plus $d(G)(\Delta G - 1) + 1$ couleurs où $d(G)$ est le diamètre de G, i.e., la plus longue distance entre 2 nœuds (la distance est le nombre d'arêtes du plus court chemin entre les 2 nœuds).

Sevastjanow (1990) est aussi l'un des premiers à s'être intéressé aux problèmes de coloration consécutive d'arêtes et on lui doit notamment la preuve de complexité établissant que le problème de savoir si un graphe général G possède une coloration consécutive est $\mathcal{NP}$-complet.

Kubale (1992) a étudié la complexité de nombreux problèmes de coloration qui diffèrent selon les restrictions que l'on impose à ceux-ci, et ce autant pour la coloration d'arêtes que pour la coloration de nœuds. Plusieurs de ces variantes font appel à une fonction de restriction $F : N \rightarrow S$ (resp. $F : E \rightarrow S$) où S est l'ensemble des sous-ensembles de $\{1, \ldots, k\}$, N l'ensemble des nœuds, E l'ensemble des arêtes et k est une borne sur le nombre de couleurs qui peuvent être utilisées. Cette fonction définit pour chaque nœud (resp. arête) du graphe un ensemble de couleurs interdites pour la coloration des nœuds (resp. arêtes). Le nombre (resp. indice) $\chi(G)$ (resp. $\chi'(G)$) est le nombre de couleurs minimum nécessaire pour colorier les nœuds (resp. arêtes) du graphe. Le nombre (resp. indice) chromatique restreint $\chi(G, F)$ (resp. $\chi'(G, F)$) est donc le nombre minimum de couleurs nécessaire pour colorier les nœuds (resp. arêtes) de G en considérant la restriction F. Voici quelques définitions:

- Le problème du nombre chromatique (NC): pour un graphe G et un entier k, est-ce que $\chi(G) \leq k$?
- Le problème du nombre chromatique restreint (NCR): pour un graphe G, une fonction de restriction F et un entier k, est-ce que $\chi(G, F) \leq k$?
- Le problème de l'indice chromatique (IC): pour un graphe G et un entier k, est-ce que $\chi'(G) \leq k$?
- Le problème de l'indice chromatique restreint (ICR): pour un graphe G, une fonction de restriction F et un entier k, est-ce que $\chi'(G, F) \leq k$?

Tableau 1.1: Complexité de certains problèmes de coloration selon Kubale (1992)

Graphe	IC	ICR	NC	NCR
Général	NPC	NPC	NPC	NPC
Complet	$O(n)$	$O(n^3\sqrt{\log n})$	$O(n^2)$	NPC
Biparti	$O(n^2)$	NPC	$O(n^2 \log n)$	NPC
Arbres	$O(n)$	$O(n \log n)$	$O(n)$	?
Étoiles	$O(n)$	$O(n)$	$O(n)$	$O(n^3\sqrt{\log n})$
Chemins	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Les principales conclusions de Kubale (1992) sont présentées dans le tableau 1.1. Celles-ci tendent à montrer que les colorations restreintes sont au moins aussi difficiles pour la coloration d'arêtes que pour la coloration des nœuds.

Hanson *et al.* (1996 et 1998) et Pyatkin (2004) ont considéré une certaine famille de graphes bipartis dit bi-réguliers. Pyatkin (2004) ainsi que Asratian et Casselgren (2006) ont notamment obtenu des conditions suffisantes pour que cette famille de graphes ait une coloration consécutive. Si la condition donnée par Pyatkin n'est pas suffisante car très spécifique, Asratian et Casselgren ont émis l'hypothèse que la leur n'est pas seulement suffisante mais aussi nécessaire.

Giaro (1997) a montré que déterminer si un graphe biparti possède une coloration consécutive peut être établi en temps polynomial si $\Delta(G) \leq 4$ et devient $\mathcal{NP}$-complet si $\Delta(G) > 4$.

Giaro *et al.* (1999) introduisent pour la première fois la notion de déficience d'un graphe telle que présentée ici. Un graphe biparti $G = (V_1, V_2; E)$ est un graphe à n processeurs si $|V_1| = n$. Les auteurs montrent que les graphes à n processeurs où $n \leq 3$ possèdent une coloration consécutive, puis ils présentent le plus petit graphe (en terme de nombre de nœuds) possédant 19 nœuds ayant une déficience

non nulle; il s'agit d'un graphe à 4 processeurs dont la déficience est 1. Les auteurs montrent que, pour une famille de graphe M_k, qui sont à 4 processeurs, la déficience augmente proportionnellement à k prouvant ainsi que la déficience d'un graphe peut être arbitrairement élevée. De plus, pour ces mêmes graphes, il montrent qu'il existe un nœud pour lequel la déficience à ce nœud est elle aussi arbitrairement élevée. Puis ils présentent une autre famille de graphes $H_{k,l}$, les graphes de Hertz, pour lesquels la déficience est de l'ordre du nombre de nœuds.

Giaro *et al.* (2001) ont amélioré les résultats obtenus par Asratian et Kamalian (1994) en établissant à $2n(G) - 4$ la valeur de $S(G)$, la borne supérieure sur le nombre maximal de couleurs pouvant être utilisées pour colorier de façon consécutive un graphe général G possédant une telle coloration. Ils généralisent ensuite aux graphes généraux un autre résultat obtenu par Asratian et Kamalian (1994) concernant les graphes bipartis. Ils obtiennent ainsi une autre borne pour $S(G)$ soit $S(G) \leq (d(G)+1)(\Delta G-1)+1$ où $d(G)$ est le diamètre de G. La borne $S(G) \leq n-1$ est connue pour être serrée. Les auteurs présentent une famille de graphes G_m pour lesquels $S(G) \leq 2n(G_m) - 2\sqrt{n(G_m)}$, ainsi $\lim_{m\to\infty} S(G_m)/n(G_m) = 2$. Ces résultats montrent que le coefficient 2 de la borne $S(G) \leq 2n(G) - 4$ ne peut être diminué et que cette borne est assez serrée. Une preuve est donnée qu'on ne peut borner le nombre minimum de couleurs $s(G)$ pour colorier consécutivement le graphe G en terme de $\Delta(G)$ (i.e., il existe une famille de graphes G_m pour laquelle $\lim_{m\to\infty} s(G_m)/\Delta(G_m) = \infty$. Les auteurs concluent leur article en établissant que certains graphes ne sont pas coloriables consécutivement.

1.7.2 Application aux problèmes d'ordonnancement

Les problèmes de coloration consécutive se prêtent bien à la modélisation de certains problèmes d'ordonnancement de machines parallèles où un ensemble de machines M

doivent opérer sur un ensemble de pièces P. Le graphe biparti $G = (N^M, N^P; E)$ résultant permet de modéliser le problème de la façon suivante: les nœuds N^M représentent les machines et les nœuds N^P les pièces. L'arête $e = (m, p) \in E$ où $m \in N^M$ et $p \in N^P$ représente le fait que la machine m doit opérer pour une unité de temps sur la pièce p. Une coloration c de ce graphe peut être interprétée comme un ordonnancement où la couleur $c(e)$ de l'arête $e = (m, p) \in E$ est associée à un nombre entier qui représente le moment où le traitement de la pièce p est effectué par la machine m. Dans le cas où l'ensemble des opérations effectuées sur une pièce et l'ensemble des traitements effectués par une machine doit se faire sans interruption, il est alors nécessaire d'obtenir une coloration consécutive du graphe G. Si une telle coloration n'est pas possible, alors on peut se donner l'objectif de minimiser le nombre de périodes d'interruption, ce qui revient à chercher à minimiser la somme des déficiences des nœuds de G. Parmi les articles concernant des problèmes semblables à celui que l'on vient de décrire, notons ceux de de Werra (1990), de Werra et Solot (1991 et 1993), de Werra *et al.* (1991), Giaro (2001), Giaro et Kubale (2004) et Kubale et Nadolski (2005).

CHAPTER 2 : ORGANISATION DE LA THÈSE

Dans cette thèse, nous considérons deux types de problèmes de coloration de graphes et un problème de fabrication d'horaires. Comme on a pu le voir dans la revue de la littérature, il est possible de modéliser des assignations à une période en attribuant des couleurs aux arêtes ou aux nœuds d'un graphe. Cette possibilité permet de modéliser certains problèmes de fabrication d'horaires comme des problèmes de coloration de graphes. L'idée unificatrice de cette thèse est d'explorer de nouvelles approches pour résoudre des problèmes de fabrication d'horaires, en particulier le problème d'attribution d'activités dans des quarts de travail et des problèmes qui sont étroitement liés à la fabrication d'horaires, tels que les problèmes de coloration par intervalle et de minimisation de la déficience d'un graphe. Pour commencer, nous présentons aux chapitres 3, 4, et 5 des heuristiques appliquées à des problèmes de coloration de graphes et quelques observations sur ces problèmes. Nous terminons en exposant les méthodes que nous avons développées dans le but de résoudre le problème d'attribution d'activités dans des quarts de travail.

Le prochain chapitre présente une méthode heuristique basée sur la méthode tabou visant à obtenir une coloration des arêtes d'un graphe ayant la plus petite déficience possible. Quelques travaux théoriques ont porté sur ce problème, lequel permet, entre autres choses, de planifier des rencontres entre divers intervenants de la façon la plus compacte possible. Ces travaux ont mené au développement de quelques algorithmes spécialisés applicables à certaines classes de graphes, mais aucun ne pouvant être appliqués pour un graphe général. C'est précisement ce que permet l'approche tabou présentée dans ce chapitre. Les voisinages de notre méthode tabou tirent profit de la nature des problèmes de coloration d'arêtes afin d'améliorer ses performances. Dans le but de comparer les qualités de ces voisinages *intelligents*, nous avons

aussi développé un voisinage plus simple qui sert de point de comparaison. Aussi, quelques résultats théoriques sont donnés, ces derniers permettent d'obtenir plus d'informations sur la solution optimale recherchée et procurent un nouvel éclairage sur le problème.

Le quatrième chapitre s'intéresse à un problème de coloration des nœuds d'un graphe, le problème de coloration par intervalle, dans lequel chaque nœud peut recevoir plus d'une couleur. Dans ce cas aussi, nous cherchons à obtenir des colorations formant des intervalles, sauf qu'ici cette contrainte ne peut être violée et l'objectif est de minimiser le nombre de couleurs utilisées. Il existe un certain nombre de problèmes de graphes présentant des similarités avec ce problème. C'est le cas notamment du problème de coloration par bande. Toutefois on peut facilement se méprendre et penser que les deux problèmes sont identiques. Ce n'est toutefois pas le cas et c'est la partie centrale du quatrième chapitre. Dans ce chapitre, nous démontrons que le problème de coloration par intervalle peut être ramené à la résolution d'un petit nombre de problèmes de coloration par bande. Nous présentons ensuite l'algorithme que nous avons développé afin d'appliquer cette réduction. Ce dernier résout les sous-problèmes de coloration par bande à l'aide d'un algorithme existant, l'algorithme de Prestwich (2002). La section d'expérimentations numériques permet de comparer l'utilité de cette réduction en la comparant à un algorithme existant pour la résolution du problème de coloration par intervalle.

Poursuivant notre étude des problèmes de coloration par intervalle, le cinquième chapitre étudie la question de savoir quand deux colorations différentes sont identiques à une permutation des couleurs près. Cette question est d'importance car on peut améliorer l'efficacité des algorithmes résolvant ce problème en évitant de visiter des solutions équivalentes. Nous présentons et démontrons une condition nécessaire et suffisante pour que deux colorations par intervalle soient identiques à une permutation des couleurs près. Pour fin de comparaison, deux algorithmes heuristiques,

basés sur la méthode tabou et inspirés de *Tabucol* (voir Hertz et de Werra 1987) ont été développés. Le deuxième algorithme est obtenu en modifiant le voisinage et la liste tabou du premier de sorte à empêcher, dans la mesure du possible, la visite de solutions équivalentes. Les expérimentations numériques montrent qu'un algorithme tabou pour ce problème peut être amélioré en empêchant la visite de solutions identiques à l'aide du critère que nous avons découvert.

Le dernier chapitre concerne la résolution du problème d'attribution d'activités à des quarts de travail. En regard au succès des méthodes heuristiques basées sur la méthode tabou pour résoudre les problèmes de coloration de graphes permettant de modéliser des problèmes de fabrication d'horaires, nous présentons d'abord une méthode tabou pour résoudre l'attribution des activités. Les instances étudiées étant très grandes et difficiles à résoudre. Des méthodes de réduction de la taille du problème, utilisant des modèles de programmation linéaire, ont été ajoutées à notre algorithme. Ces réductions ont permis de diminuer substantiellement la taille des problèmes et d'améliorer grandement l'efficacité de l'algorithme. Les résultats présentés pour cet algorithme sont satisfaisants. Cependant, cette approche n'améliore pas les méthodes existantes pour ce problème. Un modèle de programmation linéaire en nombres entiers, qui est une extension des modèles utilisées pour les méthodes de réduction, est ensuite considéré. Des résultats pour différentes variantes de ce modèle sont donnés et l'efficacité affichée par ces dernières est de beaucoup supérieure à celle obtenue avec l'approche tabou, en particulier lorsqu'il est connu qu'il existe une solution sans sous-couverture et sur-couverture.

Dans la conclusion, nous faisons une synthèse des nouvelles techniques de résolution performantes et générales proposées pour résoudre les deux problèmes de coloration de graphes étudiés ainsi que le problème d'attribution d'activités à des quarts de travail. Nous y proposons aussi quelques pistes pour de futures recherches.

CHAPITRE 3 : LOWER BOUNDS AND A TABU SEARCH ALGORITHM FOR THE MINIMUM DEFICIENCY PROBLEM

Accepté pour publication dans *Journal of Combinatorial Optimization* le 13 septembre 2007.

Lower bounds and a tabu search algorithm for the minimum deficiency problem

Mathieu Bouchard, Alain Hertz, Guy Desaulniers

École Polytechnique de Montréal

Département de Mathématiques et Génie Industriel

C.P. 6079, succ. Centre-Ville

Montréal, Québec, Canada H3C 3A7

{Mathieu.Bouchard, Alain.Hertz, Guy.Desaulniers}@gerad.ca

Abstract

An edge coloring of a graph $G = (V, E)$ is a function $c : E \to \mathbb{N}$ that assigns a color $c(e)$ to each edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' have a common endpoint. Denoting $S_v(G, c)$ the set of colors assigned to the edges incident to a vertex $v \in V$, and $D_v(G, c)$ the minimum number of integers which must be added to $S_v(G, c)$ to form an interval, the deficiency $D(G, c)$ of an edge coloring c is defined as the sum $\sum_{v \in V} D_v(G, c)$, and the span of c is the number of colors used in c. The problem of finding, for a given graph, an edge coloring with a minimum deficiency is NP-hard. We give new lower bounds on the minimum deficiency of an edge coloring and on the span of edge colorings with minimum deficiency. We also propose a tabu search algorithm to solve the minimum deficiency problem and report experiments on various graph instances, some of them having a known optimal deficiency.

3.1 Introduction

An *edge coloring* of a graph $G = (V, E)$ is a function $c \rightarrow I\!N$ that assigns a color $c(e)$ to each edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' share a common endpoint. A *matching* in G is a set of pairwise non-adjacent edges, and it is *perfect* if it covers all vertices of G (i.e, every vertex of G is incident to exactly one edge of the matching). A *chain* in G is a sequence $(v_1, \ldots, v_p)$ of vertices such that v_i and v_{i+1} are linked by an edge in G for all $i = 1, \ldots, p-1$. For a vertex $v \in V$, we denote E_v the set of edges incident to v, and $deg_G(v) = |E_v|$ its *degree*. Also, the maximum and the minimum degree in G are denoted $\Delta(G)$ and $\delta(G)$, respectively. A k-regular graph is a graph in which all vertices have degree k (i.e., $\Delta(G) = \delta(G) = k$).

For an edge coloring c, let $c_{min}(v) = \min_{e \in E_v}\{c(e)\}$ and $c_{max}(v) = \max_{e \in E_v}\{c(e)\}$ denote, respectively, the smallest and the largest color assigned to an edge incident to v. A *consecutive coloring* is an edge coloring where $c_{max}(v) = c_{min}(v) + |E_v| - 1$ for all vertices $v \in V$. In other words, an edge coloring is a consecutive coloring if and only if the colors on the edges of E_v are consecutive for every $v \in V$. The problem of determining a consecutive coloring (if any) of a graph was introduced by Asratian and Kamalian [2]. It often arises in scheduling problems with compactness constraints [5]. There are graphs with no consecutive colorings, the simplest examples being the odd cycles. The problem of determining whether or not a given graph admits a consecutive coloring is $\mathcal{NP}$-complete [14].

Given an edge coloring c, we denote $S_v(G, c)$ the set of colors assigned to the edges in E_v and $D_v(G, c)$ the minimum number of integers which must be added to $S_v(G, c)$ to form an interval. The missing integers are called the *missing colors at vertex* v. The *deficiency of an edge coloring* c of G is defined as the sum $D(G, c) = \sum_{v \in V} D_v(G, c)$. Hence, an edge coloring c is a consecutive coloring of G if and only if $D(G, c) = 0$. The *deficiency of a graph* G, denoted $Def(G)$, is the minimum deficiency $D(G, c)$ over all edge colorings c of G. This concept, which was introduced

by Giaro et al. [4], provides a measure of how close G is to have a consecutive coloring since the deficiency of a graph is the minimum number of pendant edges that must be added to G such that the resulting graph has a consecutive coloring. An edge coloring c with $D(G,c) = Def(G)$ is said *optimal*, and the problem of finding an optimal edge coloring is called the *Minimum Deficiency Problem* (MDP for short). Giaro [3] has proved that the MDP is NP-hard. Outside this result and the works on bipartite graphs ([1, 4, 8, 9, 12], k-regular graphs [13], odd cycles, wheels, almost wheels and complete graphs [6], very little is known about the deficiency of general graphs. In particular, to our knowledge, no algorithms have yet been proposed in the literature for solving the MDP.

Consider any edge coloring c, and let p and q denote the smallest and the largest color used in c, respectively. $Span(c)$ denotes its *span* which is defined as the number of different colors used in c (i.e., $Span(c) = |\cup_{v\in V} S_v(G,c)|$). If there is a color $\alpha \in \{p,\ldots,q\}$ that is not used in c, then the edge coloring c' obtained by setting $c'(e) = c(e)$ if $c(e) < \alpha$ and $c'(e) = c(e) - 1$ otherwise has a deficiency $D(G,c') \leq D(G,c)$. Therefore, in the following, we assume that all colors in $\{p,\ldots,q\}$ are used in c. Also, without loss of generality, we assume that $p = 1$, which means that $Span(c) = q$.

While Vizing [15] has proved that the edges of a graph $G = (V,E)$ can be colored using at most $\Delta(G) + 1$ colors, it may happen that $Span(c)$ is much larger than $\Delta(G)+1$ for all optimal edge colorings c. The next section provides lower bounds on the span of an optimal edge coloring and on $Def(G)$ for particular graphs G. In Section 3.3, we describe two variants of a tabu search algorithm for the MDP, one of them being especially designed for instances needing much more than $\Delta(G)+1$ colors to reach the optimal deficiency. Section 3.4 is devoted to computational experiments performed on several classes of graphs, some having a known optimal deficiency.

3.2 Lower bounds on $Span(c)$ and $Def(G)$

All results proved in this section will be very useful to measure the performance of the algorithms tested in Section 3.4. The first proposition was proved in [6] and provides a lower bound on the deficiency $Def(G)$ of k-regular graphs G with an odd number of vertices.

Proposition 1. *[6] Let G be a k-regular graph with an odd number of vertices. Then*

$$Def(G) \geq \frac{k}{2}.$$

Observe that each color used in an edge coloring c of a graph G with an odd number of vertices is missing in at least one set $S_v(G,c)$. As a consequence, graphs with an odd number of vertices typically have larger optimal deficiencies than similar ones with an even number of vertices. Their optimal edge colorings also have, in general, larger spans. For example, it is known (see [6]) that a complete graph G with an even number n of vertices admits a consecutive coloring (i.e., $Def(G) = 0$) that uses $\Delta(G) = n - 1$ colors while, for a complete graph with an odd number n of vertices, its deficiency is $Def(G) = \frac{n-1}{2}$ and the span of an optimal edge coloring c is possibly equal to $\frac{3(n-1)}{2}$. The next proposition links the deficiency of an edge coloring with its span in graphs with an odd number of vertices.

Proposition 2. *Let G be a graph with an odd number of vertices, and let c be an edge coloring of G. Then*

$$Span(c) \geq 2\delta(G) - D(G,c).$$

proof. First, observe that $c_{max}(v) \geq \delta(G)$ and $c_{min}(v) \leq Span(c) - \delta(G) + 1$ for all $v \in V$. If $Span(c) - \delta(G) + 1 > \delta(G)$, then $Span(c) \geq 2\delta(G) \geq 2\delta(G) - D(G,c)$.

Otherwise, each of the $2\delta(G)-Span(c)$ colors in $\{Span(c)-\delta(G)+1,\ldots,\delta(G)\}$ is used in c (since we assume that each color in $\{1,\cdots,Span(c)\}$ is used in c). For each color i in this set, there exists at least one vertex $v_i \in V$ such that $i \notin S_{v_i}(G,c)$ because $|V|$ is odd. Furthermore, i is a missing color at v_i since $c_{min}(v_i) < i < c_{max}(v_i)$. Consequently, there are at least $2\delta(G) - Span(c)$ missing colors at the vertices, that is, $D(G,c) \geq 2\delta(G) - Span(c)$ which is equivalent to $Span(c) \geq 2\delta(G) - D(G,c)$. □

The following result is a direct corollary of Propositions 1 and 2.

Corollary 1. *Let G be a k-regular graph with an odd number of vertices, and let c be an edge coloring of G with $D(G,c) = \frac{k}{2}$. Then*

$$Span(c) \geq \frac{3k}{2}.$$

proof. Proposition 2 with $\delta(G) = k$ and $D(G,c) = \frac{k}{2}$ gives $Span(c) \geq 2k - \frac{k}{2} = \frac{3k}{2}$. □

A result similar to Proposition 2 can be proved for graphs G with an even number of vertices and with an edge belonging to all perfect matchings in G.

Proposition 3. *Let G be a graph with an even number of vertices and with an edge e belonging to all perfect matchings in G, and let c be an edge coloring of G. Then*

$$Span(c) \geq 2\delta(G) - \frac{D(G,c)}{2} - 1.$$

proof. As in Proposition 2, $c_{max}(v) \geq \delta(G)$ and $c_{min}(v) \leq Span(c) - \delta(G) + 1$ for all $v \in V$. If $Span(c) - \delta(G) + 1 > \delta(G)$, then $Span(c) \geq 2\delta(G) \geq 2\delta(G) - \frac{D(G,c)}{2} - 1$. Otherwise, each of the $2\delta(G) - Span(c)$ colors in $\{Span(c) - \delta(G) + 1,\ldots,\delta(G)\}$

is used in the edge coloring c. Since edge e belongs to all perfect matchings in G, the edges with a color $i \neq c(e)$ do not induce a perfect matching. Hence, since $|V|$ is even, there exist for all colors $i \neq c(e)$ at least two vertices v_i and w_i in V such that $i \notin S_{v_i}(G,c) \cup S_{w_i}(G,c)$. Furthermore, if $i \in \{Span(c) - \delta(G) + 1, \ldots, \delta(G)\}$, then i is a missing color at v_i and w_i since $c_{min}(v_i) < i < c_{max}(v_i)$ and $c_{min}(w_i) < i < c_{max}(w_i)$. Consequently, there are at least $2(2\delta(G) - Span(c) - 1)$ missing colors at the vertices, that is, $D(G,c) \geq 2(2\delta(G) - Span(c) - 1)$ which is equivalent to $Span(c) \geq 2\delta(G) - \frac{D(G,c)}{2} - 1$. □

The next proposition was proved in [4]. It provides a lower bound on the minimum deficiency of a graph.

Proposition 4. *[4] Let $G = (V, E)$ be a graph, $v \in V$ a vertex with $deg_G(v) > 1$, and $p_1, \ldots, p_\ell$ and $q_1, \ldots, q_\ell$ two sequences of ℓ positive integers. Assume that, for any two vertices $u \neq w$ adjacent to v there is a chain $(u = v_1, v_2, \ldots, v_{p_i+1} = w)$ connecting u and w in G, where p_i is one of the numbers of the first sequence, such that*

$$deg_G(v) + p_i \geq q_i + \sum_{j=1}^{p_i+1} deg_G(v_j).$$

Then $Def(G) \geq \min\{q_1, \ldots, q_\ell\}$.

The *distance* between two vertices u and v in a graph G is the number of edges in a shortest chain connecting them. It is denoted $d_G(u,v)$. The *diameter* of a graph G, denoted $Diam(G)$, is the maximum distance between any two of its vertices. For a graph $G = (V, E)$ and a vertex $v \in V$, we denote $G - v$ the graph obtained from

G by removing v and all edges incident to v. The next two results are corollaries of Proposition 4.

Corollary 2. *Let $G = (V, E)$ be a graph, q a positive integer, and $v \in V$ a vertex with $deg_G(v) > 1$ such that $G - v$ is connected. Assume that for any two vertices $u \neq w$ adjacent to v and for any shortest chain $(u = v_1, v_2, \ldots, v_{d_{G-v}(u,w)+1} = w)$ connecting u and w in $G - v$, we have*

$$deg_G(v) + d_{G-v}(u, w) \geq q + \sum_{j=1}^{d_{G-v}(u,w)+1} deg_G(v_j).$$

Then $Def(G) \geq q$.

proof. Define $p_1, \ldots, p_\ell$ as the sequence of the distances in $G - v$ between all pairs of vertices $u \neq w$ adjacent to v, and define $q_i = q$ for $i = 1, \ldots, \ell$. Consider any two vertices $u \neq w$ adjacent to v and let $(u = v_1, v_2, \ldots, v_{d_{G-v}(u,w)+1} = w)$ be a shortest chain connecting u and w in $G - v$. There is a number p_i in the sequence such that $d_{G-v}(u, w) = p_i$, and the inequality $deg_G(v) + d_{G-v}(u, w) \geq q + \sum_{j=1}^{d_{G-v}(u,w)+1} deg_G(v_j)$ can therefore be rewritten as

$$deg_G(v) + p_i \geq q_i + \sum_{j=1}^{p_i+1} deg_G(v_j).$$

Since this inequality is valid for any two vertices $u \neq w$ adjacent to v, it follows from Proposition 4 that $Def(G) \geq \min\{q_1, \ldots, q_\ell\} = q$. □

Corollary 3. *Let $G = (V, E)$ be a graph containing a vertex v with $deg_G(v) = |V| - 1$ and such that $G - v$ is connected and k-regular. Then*

$$Def(G) \geq |V| - kDiam(G - v) - k - 2.$$

proof. Consider any two vertices $u \neq w$ adjacent to v and let $(u = v_1, \ldots, v_{d_{G-v}(u,w)+1} = w)$ be any shortest chain connecting u and w in $G - v$. For $q = |V| - kDiam(G - v) - k - 2$, we have

$$\begin{aligned} q &= |V| - kDiam(G - v) - k - 2 \\ &\leq |V| - kd_{G-v}(u, w) - k - 2 \\ &= |V| - 1 + d_{G-v}(u, w) - (d_{G-v}(u, w) + 1)(k + 1) \\ &= deg_G(v) + d_{G-v}(u, w) - \sum_{i=1}^{d_{G-v}(u,w)+1} deg_G(v_i) \end{aligned}$$

The conclusion follows from Corollary 2. □

3.3 A tabu search algorithm for the MDP

Let S be the set of solutions to a combinatorial optimization problem, and f a function to be minimized over S. For a solution $s \in S$, let $N(s)$ denote the *neighborhood* of s which is defined as the set of solutions in S obtained from s by performing a local change, called *move*. A local search is an algorithm that generates a sequence $s_0, s_1, \ldots, s_r$ of solutions in S, where s_0 is an initial solution and each s_i $(i > 1)$ belongs to $N(s_{i-1})$. Tabu search is one of the most famous local search algorithms. In

order to avoid cycling, tabu search uses a *tabu list* T that contains forbidden moves. Hence, a move m from s_{i-1} to s_i can only be performed if m does not belong to the tabu list T, unless $f(s_i) < f(s^*)$, where s^* is the best solution encountered so far. The general scheme of a tabu search algorithm is given in Figure 3.1. For more details on tabu search, the reader may refer to [7].

Tabu search

Generate an initial solution $s \in S$, set $T \leftarrow \emptyset$ and $s^* \leftarrow s$;

while no stopping criterion is met **do**

Determine a solution $s' \in N(s)$ with minimum value $f(s')$ such that the move from s to s' does not belong to T or $f(s') < f(s^*)$;

if $f(s') < f(s^*)$ **then**

set $s^* \leftarrow s'$;

end if

Set $s \leftarrow s'$ and update T.

end while

Figure 3.1: General scheme of a tabu search algorithm.

For the MDP, we define S as the set of edge colorings of G, while the value $f(c)$ of a solution $c \in S$ is its deficiency $D(G, c)$. Vizing [15] has proved that the edges of a graph $G = (V, E)$ can be colored using at most $\Delta(G) + 1$ colors. The proof of Vizing's theorem is constructive and can be turned into an $O(|V||E|)$ algorithm. We use this algorithm to build an initial solution for our adaptation of tabu search to the

MDP. We consider two types of neighborhoods, a simple one and a more complex one, which are defined in the two next subsections.

3.3.1 A simple neighborhood

The first proposed neighborhood, denoted $N_1(c)$, contains edge colorings obtained from c by changing the color of a single edge e. More precisely, given an edge coloring c, a neighbor in $N_1(c)$ is obtained by choosing an edge e and assigning a new color to e that does not belong to $S_u(G,c) \cup S_v(G,c)$, where u and v are the endpoints of e. In order to try to reduce $D_u(G,c)$ and $D_v(G,c)$, the new color should preferably be chosen in $\{c_{min}(u), \ldots, c_{max}(u)\} \cup \{c_{min}(v), \ldots, c_{max}(v)\}$. We also consider the possibility of assigning color $c_{min}(u) - 1, c_{max}(u) + 1, c_{min}(v) - 1$ or $c_{max}(v) + 1$ to e. Notice that at least two of these four colors, namely $\min\{c_{min}(u) - 1, c_{min}(v) - 1\}$ and $\max\{c_{max}(u) + 1, c_{max}(v) + 1\}$, do not belong to $S_u(G,c) \cup S_v(G,c)$, and it may even happen that $D_u(G,c)$ decreases when assigning one of these four colors to e. For example, if $S_u(G,c) = \{2,4\}$ and $c(e) = c_{max}(u) = 4$, then $D_u(G,c) = 1$, and by assigning color $c_{min}(u) - 1 = 1$ to e one gets $S_u(G,c) = \{1,2\}$, which means that $D_u(G,c)$ decreases to 0. We therefore define $N_1(c)$ as the set of edge colorings that are obtained from c by assigning a new color to an edge e with endpoints u and v, this new color being chosen in $(\{c_{min}(u) - 1, \ldots, c_{max}(u) + 1\} \cup \{c_{min}(v) - 1, \ldots, c_{max}(v) + 1\}) \setminus (S_u(G,c) \cup S_v(G,c))$. As shown in the next proposition, it is possible to reach every solution in S starting from any edge coloring c, and only using the neighborhood N_1.

Proposition 5. *Given two distinct edge colorings c and c', there exists a finite sequence $c_1, \ldots, c_k$ of edge colorings such that $c = c_1$, $c' = c_k$ and each c_{i+1} belongs to $N_1(c_i)$.*

proof. Consider two distinct edge colorings c and c' of $G = (V,E)$, and denote $h(c,c')$

the number of edges $e \in E$ such that $c(e) \neq c'(e)$. It is sufficient to prove that there exists a finite sequence $c_1, \ldots, c_r$ of edge colorings such that $c = c_1$, each c_{i+1} belongs to $N_1(c_i)$, and $h(c_r, c') < h(c, c')$. This is sufficient because the sought sequence $c_1, \ldots, c_k$ can be seen as a concatenation of such sequences. Let e be an edge with endpoints u and v such that $c(e) \neq c'(e)$. Using neighborhood N_1, at most two moves are necessary to transform c into an edge coloring $\tilde{c}$ with $h(\tilde{c}, c') \leq h(c, c')$ and $c'(e) \notin S_u(G, \tilde{c}) \cup S_v(G, \tilde{c})$. Indeed, if $c'(e) \in S_u(G, c)$, then there is an edge e_u with endpoints u and $w \neq v$ such that $c(e_u) = c'(e)$. Because $c'(e_u) \neq c'(e)$, the neighbor c'' of c obtained by assigning color $\max\{c_{max}(u), c_{max}(w)\} + 1$ to e_u satisfies $h(c'', c') \leq h(c, c')$. Similarly, one move is sufficient to remove color $c'(e)$ from $S_v(G, c)$ (if needed). So let $\tilde{c}$ be an edge coloring with $\tilde{c}(e) \neq c'(e)$, $c'(e) \notin S_u(G, \tilde{c}) \cup S_v(G, \tilde{c})$, and $h(\tilde{c}, c') \leq h(c, c')$. It is now sufficient to prove that, using neighborhood N_1, there is a sequence of moves that transforms $\tilde{c}$ into an edge coloring $\tilde{c}'$ with $h(\tilde{c}', c) = h(\tilde{c}, c) - 1 < h(c', c)$. Let $\alpha = \min\{\tilde{c}_{min}(u), \tilde{c}_{min}(v)\}$ and $\beta = \max\{\tilde{c}_{max}(u), \tilde{c}_{max}(v)\}$.

- If $\alpha < c'(e) < \beta$, $\tilde{c}'$ is obtained from $\tilde{c}$ by assigning color $c'(e)$ to e.
- If $c'(e) < \alpha$, define c_t, for $t = 1, \ldots, \alpha - c'(e)$, such that $c_t(e') = \tilde{c}(e')$ for all $e' \neq e$, and $c_t(e) = \alpha - t$. We have $c_1 \in N_1(\tilde{c})$, $c_t \in N_1(c_{t-1})$ for all $t \in \{2, \ldots, \alpha - c'(e)\}$, and $h(c_{\alpha - c'(e)}, c') = h(\tilde{c}, c) - 1$. Hence, $\tilde{c}' = c_{\alpha - c'(e)}$.
- If $\beta < c'(e)$, define c_t, for $t = 1, \ldots, c'(e) - \beta$, such that $c_t(e') = \tilde{c}(e')$ for

all $e' \neq e$, and $c_t(e) = \beta + t$. We have $c_1 \in N_1(\tilde{c})$ and $c_t \in N_1(c_{t-1})$ for all $t \in \{2, \ldots, c'(e) - \beta\}$, and $h(c_{c'(e)-\beta}, c') = h(\tilde{c}, c) - 1$. Hence, $\tilde{c}' = c_{c'(e)-\beta}$.

□

3.3.2 A more complex neighborhood

Given an edge coloring c, a vertex u, an edge e incident to u, and a color $\alpha \notin S_u(G, c)$ with $c_{min}(u) - 1 \leq \alpha \leq c_{max}(u) + 1$, consider the partial graph of G containing only those edges with color $c(e)$ or α, and let $B(c, u, e, \alpha)$ be the connected component of this subgraph containing e. $B(c, u, e, \alpha)$ is a chain with endpoints u and v, with $|S_v(G, c) \cap \{c(e), \alpha\}| = 1$. A neighbor of c is obtained by permuting colors $c(e)$ and α in $B(c, u, e, \alpha)$. Such a move is called a *bichromatic exchange*. Note that the deficiency of the internal vertices of the chain $B(c, u, e, \alpha)$ does not change. Such moves are illustrated in Figure 3.2 in which the numbers correspond to the colors. Because $S_u(G, c) = \{2, 3, 5\}$, the possible choices for α are $1, 4$ and 6. The three possible neighbors are represented in Figure 3.2, with the corresponding chains in bold lines.

Notice that the simpler neighborhood N_1 is a special case of this more complex neighborhood where $B(c, u, e, \alpha)$ contains only one edge e with endpoints u and v and $\alpha \notin S_v(G, c)$. For example, the first neighbor in Figure 3.2 is also contained in the first neighborhood. Denoting $N_2(c)$ the set of neighbor edge colorings that can be obtained with such moves on c, we have the following corollary of Proposition 5.

Corollary 4. *Given two edge colorings c and c', there exists a sequence $c_1, \ldots, c_k$ of edge colorings such that $c = c_1$, $c' = c_k$ and each c_{i+1} belongs to $N_2(c_i)$.*

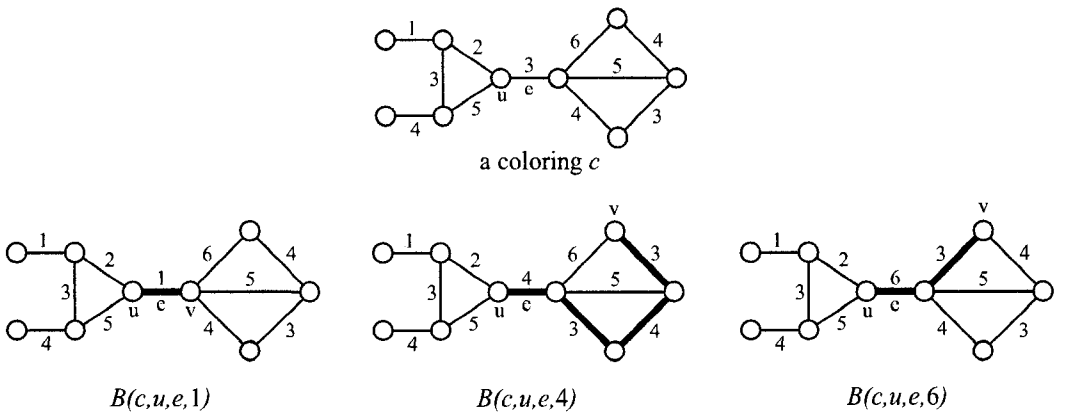


Figure 3.2: Illustration of the second neighborhood.

3.3.3 Tabu list and stopping criteria

When moving from c to c' with neighborhood N_1, we change the color of a single edge e, and this edge is introduced into the tabu list T with the meaning that it is forbidden to change the color of e for the next I_{max} iterations (where I_{max} is a parameter). For the second neighborhood N_2, a neighbor of c is obtained by performing a bichromatic exchange on a chain P with extreme edges e_1 and e_2. We introduce both edges e_1 and e_2 into the tabu list with the meaning that it is forbidden for the next I_{max} iterations to perform a bichromatic exchange on a chain if one of its extreme edges is in the tabu list. When a move from an edge coloring c to a neighbor one c' is tabu, we say that edge coloring c' is tabu.

Given any lower bound L on $Def(G)$, the algorithm is stopped as soon as an edge coloring c is reached with $D(G,c) = L$. Notice that it is always possible to set L equal to 0, while better lower bounds are known for particular classes of graphs (see Section 3.2). Another stopping criterion is also considered, namely, a time limit of 20 minutes.

3.3.4 The proposed algorithms

Our first idea was to develop an algorithm based only on the neighborhood N_2 since it includes the neighborhood N_1. However, as mentioned above, the initial solution uses at most $\Delta(G) + 1$ colors and, for some classes of graphs (see Section 3.2), it is proved that more colors are needed to obtain an optimal deficiency. Preliminary experiments have shown that the number of colors does not vary very much when using neighborhood N_2. This is probably due to the fact that the creation of a new color (which is also a move using N_1) typically increases the deficiency of the edge coloring. Note however that it may happen that an increase in the number of colors decreases the deficiency, as illustrated with the graph in Figure 3.3, where an edge coloring c with 3 colors and $D(G, c) = 1$ is transformed into an edge coloring $c' \in N_1(c)$ with 4 colors and $D(G, c) = 0$.

Figure 3.3: An increase in the number of colors may decrease the deficiency.

In summary, $N_2(c)$ contains neighbors that increase the span, but moves from c to such a neighbor are rarely performed by a tabu search algorithm since they typically increase the deficiency. Since such moves are in N_1 and are necessary to increase the number of colors used, we force the algorithm to perform moves using N_1 on a regular basis. More precisely, every t iterations (where t is a parameter), we only consider neighbors in $N_1(c)$ instead of $N_2(c)$.

To generate a neighbor c' of c, we construct the set A of neighbor solutions c' which are not tabu or such that $D(G, c') < D(G, c^*)$, where c^* is the current best edge coloring. We use a first improvement strategy. More precisely, we first label

the vertices from 1 to $|V|$, in a random order, and then order the edges so that the edge e with endpoints u and v precedes the edge e' with endpoints u' and v' if $\min\{u,v\}<\min\{u',v'\}$ or $\min\{u,v\}=\min\{u',v'\}$ and $\max\{u,v\}<\max\{u',v'\}$. We then scan the edges according to this ordering, and for every edge e with endpoints $u < v$, we do the following:

- when exploring $N_1(c)$, we consider every color α in $(\{c_{min}(u)-1,\ldots,c_{max}(u)+1\}\cup\{c_{min}(v)-1,\ldots,c_{max}(v)+1\})\setminus(S_u(G,c)\cup S_v(G,c))$, in increasing order, and evaluate the neighbor of c obtained by assigning color α instead of $c(e)$ to e.
- when exploring $N_2(c)$, we consider every color α in $I = \{c_{min}(u)-1,\ldots,c_{max}(u)+1\}\setminus S_u(G,c)$, in increasing order, and evaluate the neighbor of c obtained by permuting colors $c(e)$ and α in $B(c,u,e,\alpha)$. We then also consider every color α in $\{c_{min}(v)-1,\ldots,c_{max}(v)+1\}\setminus(S_v(G,c)\cup I)$, in increasing order, and evaluate the neighbor of c obtained by permuting colors $c(e)$ and α in $B(c,v,e,\alpha)$.

If a solution c' with $D(G,c') < D(G,c)$ is encountered during this exploration, we stop the process and choose c' as a neighbor of c. Otherwise, when the whole set A is known, we choose its best member c' as a neighbor of c. The proposed tabu search algorithm for the MDP, which we call TabuMDP, is summarized in Figure 3.4.

Computational experiments which are reported in the next section will show that the use of N_1 every t iterations is sometimes not sufficient to increase the number of colors. We have therefore implemented a variation of the above algorithm, called TabuMDP*, in which each neighbor c' in the neighborhood $N_2(c)$ of the current solution c is evaluated according to the following function f that differs slightly from $D(G,c')$ when c uses at most $Span(c^*)+1$ colors:

$$f(G,c') = \begin{cases} D(G,c') - b & \text{if } Span(c) \leq Span(c^*)+1 \text{ and } Span(c') > Span(c^*) \\ D(G,c') & \text{otherwise.} \end{cases}$$

TabuMDP

Generate an initial edge coloring $c \in S$ using Vizing's algorithm;

Set $T \leftarrow \emptyset$, $c^* \leftarrow c$ and $i \leftarrow 0$ (iteration counter);

while no stopping criterion is met **do**

if t divides i **then** set $\ell \leftarrow 1$ **else** set $\ell \leftarrow 2$;

Let A be the set of solutions $c' \in N_\ell(c)$ such that c' is not tabu or $D(G, c') < D(G, c^*)$;

if A contains a solution c' with $D(G, c') < D(G, c)$ **then** select one such solution according to the above first improvement strategy **else** determine at random a solution $c' \in A$ with minimum value $D(G, c')$;

if $D(G, c') < D(G, c^*)$ **then** set $c^* \leftarrow c'$;

Set $c \leftarrow c'$, $i \leftarrow i + 1$ and update T.

end while

Figure 3.4: A tabu search algorithm for the MDP.

This means that if the current solution c uses at most one more color than the current best edge coloring c^*, then all edge colorings $c' \in N_2(c)$ with more colors than c^* get a bonus b, which is a parameter of TabuMDP*. Function f is only used to make a choice for the next solution to be visited when no neighbor c' has a deficiency $D(G, c')$ strictly smaller than $D(G, c)$. The usual function $D(G, c')$ is used to update c^*. More precisely, the above choice of a neighbor c' of c is replaced by

Let A be the set of solution $c' \in N_\ell(c)$ such that c' is not tabu or $D(G,c') < D(G,c^*)$;
if A contains a solution c' with $D(G,c') < D(G,c)$ **then** select one such solution according to the above first improvement strategy **else** determine a solution $c' \in A$ with minimum value $f(G,c')$;

As already mentioned, the number of colors needed to reach an optimal deficiency for graphs with an odd number of vertices is typically much larger than $\Delta(G)+1$. TabuMDP* is therefore particularly useful for these graphs. Note that, as shown in Proposition 3, the span of an optimal edge coloring of a graph with an even number of vertices is also possibly much larger than $\Delta(G)+1$. For this reason, and for the sake of completeness, we will report results obtained with TabuMDP* on all graphs, including those with an even number of vertices.

3.4 Computational experiments

To our knowledge, no algorithms have been yet developed for the MDP so that it is impossible to compare our tabu search algorithms with an existing algorithm. For this reason, we have decided to perform computational experiments on special classes of graphs for which good lower bounds on the minimum deficiency are known. We also consider random graphs and graphs for which reaching a minimum deficiency requires the use of an edge coloring c with $Span(c)$ much larger than $\Delta(G)+1$. A subsection is devoted to each one of the seven considered graph classes. Also, as already mentioned, if often happens that graphs with an odd number of vertices have a larger deficiency than graphs in the same class but with an even number of vertices. We therefore consider graphs with an odd number of vertices as more challenging for the MDP, and this explains "strange" numbers of vertices used in our experiments (e.g., 501 instead of 500 vertices).

The parameters of the algorithms were selected on the basis of preliminary experiments. In TabuMDP, parameter t (that imposes the use of N_1 instead of N_2) is set to 100. To evaluate the advantage of using N_2 instead of N_1, we also report results obtained with $t = 1$ which means that all moves are made using neighborhood N_1. The bonus b used in TabuMDP* is set to 10, and the tabu list length I_{max} is set to $\sqrt{|E|}$. Other parameters have sometimes produced slightly better results for some particular instances. Hence, a better tuning of the parameters would certainly improve the reported results. However, since a relatively large set of values were tested for each parameter, we are confident that the general behavior of the proposed tabu search algorithms would not change with optimised parameters.

All tests were performed with a time limit of 20 minutes on an AMD Opteron(tm) 285/2.6 GHz Processor. Note however that an algorithm may stop sooner when a known lower bound L on $Def(G)$ is reached.

3.4.1 Random graphs

The first experiments are made on random graphs with edge density $d = 0.5$. In these graphs, each pair of vertices is linked by an edge with probability 0.5, independently for each pair. We consider random graphs with 125, 250, 501 and 1000 vertices. For each graph size, we have generated four instances. Results are reported in Table 3.1 using TabuMDP with $t = 100$ and $t = 1$, as well as TabuMDP* with $t = 100$. The first three columns contain the name of the instance, its number of vertices, and its number of edges. For each algorithm, we report the best found deficiency $D(G, c)$, the number of colors $Span(c)$ in the corresponding edge coloring c, and the number of iterations performed within 20 minutes.

These results clearly show that the use of N_2 makes it possible to reach solutions of much better quality than those obtained using only N_1 (i.e., with $t = 1$). While the best found deficiency increases with the number of vertices when using $t = 1$, a

Table 3.1: Results for random graphs

Instance	$\|V\|$	$\|E\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$		
			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations
Rand125a	125	3803	19	81	34746491	12	105	8762432	73	73	11915501
Rand125b	125	3965	11	85	24227167	13	111	8285132	72	77	11915501
Rand125c	125	3844	15	84	29333099	13	107	8665763	69	75	11780619
Rand125d	125	3929	19	85	29991502	14	109	8754145	70	79	11265903
Rand250a	250	15452	2	148	4267571	12	179	1949200	391	148	10343194
Rand250b	250	15564	1	146	4257442	12	181	2073001	488	146	10343194
Rand250c	250	15456	1	143	4482417	13	181	1864557	547	143	15242279
Rand250d	250	15506	1	146	4392651	11	177	2165681	605	147	16869638
Rand501a	501	62747	146	291	3770866	143	320	3556031	2030	287	5613871
Rand501b	501	62431	142	289	3771975	136	315	3660843	1934	283	6645562
Rand501c	501	62637	145	287	3798399	140	315	3607125	1658	281	6201459
Rand501d	501	62718	140	290	3790899	148	315	3658384	1807	283	6045927
Rand1000a	1000	249452	19	549	190741	32	570	174992	6657	548	1260185
Rand1000b	1000	250128	22	555	187537	36	577	179784	7353	554	976954
Rand1000c	1000	249795	21	557	192601	27	575	178841	7293	555	980575
Rand1000d	1000	249604	13	548	187414	32	570	167804	7006	546	903855

clear decrease happens with $t = 100$ when moving from 125 to 250 vertices, or from 501 to 1000. As mentioned above, such a decrease can be explained by the fact that graphs with an even number of vertices probably have a smaller deficiency.

By comparing the columns $Span(c)$ for TabuMDP and TabuMDP* with $t = 100$, we can observe that TabuMDP* systematically produces edge colorings with a larger span. While this can be useful for graphs with an odd number of vertices, we observe that TabuMDP* is not competitive with TabuMDP when $|V|$ is even. TabuMDP* with $t = 100$ is however much more effective than TabuMDP with $t = 1$, even when the graphs have an even number of vertices.

Note also that, for TabuMDP with $t = 1$, the number of iterations performed within 20 minutes is possibly larger for the graphs with 250 vertices than those with 125 vertices. This is due to the fact that instead of choosing the best neighbor of the current edge coloring c, we select the first improving neighbor c' (if any). Indeed, we have observed that graphs with an even number of vertices typically have a larger proportion of improving moves than those with an odd number of vertices, and this helps reducing the time needed to perform an iteration.

While the average degree of a random graph $G = (V, E)$ with edge density $d = 0.5$ is $\frac{|V|-1}{2}$, the minimum degree $\delta(G)$ is possibly much smaller. Indeed, we know from Proposition 2 that $Span(c) \geq 2\delta(G) - D(G, c)$ for graphs with an odd number of vertices. Hence, for example, we have found an edge coloring c of Rand125a with $D(G, c) = 19$ and $Span(c) = 81$, which means that $\delta(G) \leq 50 < 62 = \frac{|V|-1}{2}$. For comparison, the next section contains results for random k-regular graphs with $\delta(G) = k = \left\lceil \frac{|V|-1}{2} \right\rceil$.

3.4.2 Random k-regular graphs

The second class of graphs is the set of k-regular graphs. These graphs have the property that each vertex is incident to exactly k edges (hence $k = \Delta(G) = \delta(G)$).

In order to randomly generate such graphs for a fixed number of vertices and a fixed k, we use the algorithm proposed by Kim and Vu [11]. To make a comparison with the graphs tested in the preceding section, we have generated four 62-regular graphs with 125 vertices, four 125-regular graphs with 251 vertices, four 250-regular graphs with 501 vertices, and four 500-regular graphs with 1000 vertices. All these graphs have an edge density approximately equal to 0.5, as in the preceding tests. Proposition 1 states that $\frac{k}{2}$ is a lower bound on $Def(G)$ if G is a k-regular graph G with an odd number of vertices. Also, Corollary 1 states that if c is an edge coloring of a k-regular graph with an odd number of vertices and $D(G,c) = \frac{k}{2}$, then c uses at least $\frac{3k}{2}$ colors. For k-regular graphs with an even number of vertices, the trivial lower bound $L = 0$ on $Def(G)$ is possibly reached (the simplest example being the graph with a single edge), while $Def(G)$ can be strictly larger than 0, as it will be shown in Section 3.4.4.

Results for these graphs are reported in Table 3.2. The columns have the same meaning as in Table 3.1, except that we mention k instead of $|E|$ (which means that $|E| = \frac{k|V|}{2}$). We have added a column, labeled "L", containing the above mentioned lower bound on the optimal deficiency. An asterisk in a column "$D(G,c)$" means that the lower bound L is reached and the MDP has therefore been solved to optimality.

We observe that TabuMDP with $t = 100$ has determined consecutive colorings (i.e., edge colorings c with $D(G,c) = 0$) for all graphs having an even number of vertices. Since the lower bound L is reached in these cases, the algorithm does not need to run for 20 minutes, and this explains the small numbers of iterations when compared to the other instances. While TabuMDP* has not determined any consecutive coloring, it is systematically better than TabuMDP on all instances with an odd number of vertices. Optimal solutions are even found for all graphs with 125 vertices. This is certainly due to the larger span of the edge colorings produced by TabuMDP*. Indeed, we can observe that when G has an odd number of vertices, TabuMDP with $t = 100$ systematically produces edge colorings c with $Span(c) =$

Table 3.2: Results for random k-regular graphs

Instance	$\|V\|$	k	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			L
			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
Reg125a	125	62	48	76	42133037	31*	106	9652055	61	67	11791857	31
Reg125b	125	62	48	76	41798804	31*	106	4221359	62	68	11809216	31
Reg125c	125	62	49	75	41628433	31*	106	10779356	60	67	11800838	31
Reg125d	125	62	48	76	42090117	31*	104	1573373	60	68	11786562	31
Reg250a	250	125	0*	129	8929	9	145	32988258	96	130	4293290	0
Reg250b	250	125	0*	131	6911	9	144	33013176	98	131	4275951	0
Reg250c	250	125	0*	131	10520	11	146	35390073	102	131	4281786	0
Reg250d	250	125	0*	131	8867	9	144	37002495	92	132	4263897	0
Reg501a	501	250	240	260	6451614	233	277	5263981	299	257	901864	125
Reg501b	501	250	239	261	6203352	232	275	5136071	301	257	884702	125
Reg501c	501	250	239	261	6320236	230	276	5942112	294	256	900619	125
Reg501d	501	250	239	261	6221520	235	279	5458658	298	257	899111	125
Reg1000a	1000	500	0*	509	100744	16	525	6045627	528	507	329584	0
Reg1000b	1000	500	0*	507	76184	25	523	6074458	522	507	337950	0
Reg1000c	1000	500	0*	507	89607	17	523	6019172	512	507	338818	0
Reg1000d	1000	500	0*	506	161824	9	517	6332346	530	508	270872	0

$2k - D(G,c)$. According to Proposition 2, this is a lower bound on the number of colors needed to reach deficiency $D(G,c)$. Hence, in order to decrease $D(G,c)$, the algorithm has to increase the span of c, and TabuMDP* clearly better succeeds in this task.

For comparison, the use of N_1 only (i.e., TabuMDP with $t = 1$) does not permit to obtain any consecutive coloring, and the best found deficiency seems to increase linearly with the number of vertices.

3.4.3 Complete graphs

Let $K(p)$ denote the complete graph with p vertices. It is a k-regular graph with $k = p - 1$. Giaro et al. [6] proved that there exists an edge coloring c of $K(p)$ with $Span(c) = \frac{3(p-1)}{2}$ and $D(K(p),c) = \frac{p-1}{2} = Def(K(p))$ if p is odd, and $Span(c) = p-1$ and $D(K(p),c) = 0 = Def(K(p))$ if p is even. Optimal edge colorings of complete graphs cannot use less colors because $Span(c) \geq \frac{3(p-1)}{2}$ when $D(K(p),c) = \frac{p-1}{2}$ and p is odd (see Corollary 1), while $Span(c)$ is obviously at least equal to $\Delta(K(p)) = p-1$ for all graphs $K(p)$, including $(p-1)$-regular graphs with an even p. Notice that optimal edge colorings of complete graphs eventually use more colors. For illustration, Figure 3.5 contains two consecutive colorings c of $K(4)$; one edge coloring uses 3 colors while the other uses 4 colors.

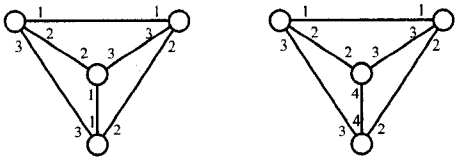


Figure 3.5: Two optimal edge colorings of the complete graph with 4 vertices.

Results for complete graphs with 25, 50, 75, 100, 125, and 150 vertices are reported in Table 3.3. The columns have the same meaning as in the previous tables, except that we indicate the optimal deficiency $Def(G)$ rather than a lower bound L.

Table 3.3: Results for complete graphs

Instance	$\|V\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			$Def(G)$
		$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
$K(25)$	25	14	34	195634551	12*	39	658087	17	31	78930677	12
$K(50)$	50	0*	54	300	0*	61	5284963	6	54	37398639	0
$K(75)$	75	59	89	59397297	37*	120	3495460	67	81	12120442	37
$K(100)$	100	0*	106	1546	4	120	53932805	30	104	12338700	0
$K(125)$	125	108	140	30142790	63	196	25143931	117	131	4789060	62
$K(150)$	150	0*	155	7484	5	168	31512675	53	154	6299412	0

We observe that TabuMDP with $t = 100$ does not produce any optimal deficiency for complete graphs with an odd number of vertices. This is probably due to the fact that the number of colors needed to reach an optimal solution is much larger than $\Delta(G)+1$. For example, for the complete graph $K(75)$, we know that all optimal edge colorings use at least 111 colors while the best solution found by TabuMDP with $t = 100$ uses only 89 colors. For comparison, TabuMDP* with $t = 100$ has found an optimal edge coloring that uses 120 colors.

As in the preceding section, it is also obvious that complete graphs with an even number of vertices are much easier to color than those with an odd number of vertices. For example, while several millions of iterations are needed by TabuMDP* to optimally color $K(75)$, TabuMDP with $t = 100$ finds an optimal solution of $K(100)$ in only 1546 iterations. Again, this can easily be explained by the small number of colors needed to reach an optimal edge coloring of complete graphs with an even number of vertices. TabuMDP with $t = 1$ also produces smaller deficiencies for graphs with an even number of vertices, but the results are not as good as those obtained with $t = 100$.

3.4.4 Schwartz's graphs

In a recent paper, Schwartz [13] has defined a family of k-regular graphs with high deficiency. These graphs are defined as follows. Let $k \geq 3$ be an odd integer, and let G be the graph obtained by subdividing an edge in the complete graph $K(k+1)$ with a vertex w. Take $\frac{k-1}{2}$ copies of G and aggregate their vertices w into a single vertex to obtain a graph H with $\frac{(k+1)(k-1)}{2}+1$ vertices. Let H_1 and H_2 be two copies of H with $w_i \in V(H_i)$ corresponding to w for $i = 1, 2$. Let $S(k)$ denote the graph obtained by making the disjoint union of H_1 and H_2 and adding an edge between w_1 and w_2. Then $S(k)$ is a k-regular graph with k^2+1 vertices, and it is proved in [13] that $Def(S(k)) \geq k-1$. For illustration, the graph $S(5)$ is given in Figure 3.6.

The lower bound $k-1$ on $Def(S(k))$ can in fact always be reached as shown in the following proposition and illustrated in Figure 3.6 for $k = 5$. We also prove that the number of colors needed to reach such an optimal deficiency is much larger than $\Delta(S(k)) = k$.

Proposition 6. *Let $k \geq 3$ be an odd integer and denote $S(k)$ its corresponding Schwartz's graph. Then,*

(a) $Def(S(k)) = k-1$, and

(b) $Span(S(k), c) \geq \frac{3k-1}{2}$ for all optimal edge colorings c of $S(k)$, and this bound is sharp.

proof. Because $Def(S(k)) \geq k-1$, we prove (a) by exhibiting an edge coloring c with $D(S(k), c) = k-1$. As mentioned in the preceding section, because k is odd, the edges of the complete graph $K(k+1)$ can be colored with k colors and without creating any deficiency. Consider $\frac{k-1}{2}$ copies $G_1, \ldots, G_{\frac{k-1}{2}}$ of $K(k+1)$ and let c_i be an edge coloring of G_i such that $D(G_i, c_i) = 0$, and c_i uses colors $i, \ldots i+k-1$.

Let G'_i be the graph obtained from G_i by subdividing the edge of color i with a vertex w, and let c'_i be the edge coloring of G'_i obtained from c_i by giving colors i and $i+k$ to the two edges incident to w, and setting $c'_i(e) = c_i(e)$ for all the other edges e in G'_i. It follows that $D_v(G'_i, c'_i) = 0$ for all vertices $v \neq w$ in G'_i. Now let H be the graph obtained by merging all graphs G'_i at vertex w, and denote c_H the resulting edge coloring. Vertex w is now adjacent to colors $1, \ldots, \frac{k-1}{2}, k+1, \ldots k+\frac{k-1}{2}$. Hence, $D(H, c_H) = D_w(H, c_H) = (k+1) - \frac{k-1}{2} - 1 = \frac{k+1}{2}$. The graph $S(k)$ is obtained by making the disjoint union of two copies H_1 and H_2 of H with $w_i \in V(H_i)$ corresponding to w for $i = 1, 2$, and adding an edge between w_1 and w_2. Consider the edge coloring c of $S(k)$ obtained by coloring H_1 and H_2 as defined above for H, and assigning any color in $\{\frac{k-1}{2} + 1, \ldots k\}$ to the edge linking w_1 and w_2. We then have $D(S(k), c) = 2\left(\frac{k+1}{2}\right) - 2 = k - 1$, which means that $Def(S(k)) \leq k - 1$.

To prove (b), notice first that the edge between w_1 and w_2 that links H_1 with H_2 belongs to all perfect matchings in $S(k)$ since both H_1 and H_2 have an odd number of vertices. Since $\delta(S(k)) = k$, it follows from Proposition 3 that $Span(c) \geq 2k - \frac{Def(S(k))}{2} - 1 = \frac{3k-1}{2}$ for all optimal edge colorings c of $S(k)$. This bound is sharp since the above edge coloring of $S(k)$ uses $k + \frac{k-1}{2} = \frac{3k-1}{2}$ colors. □

We have tested Schwartz's graphs for k =20, 33, 41, 49, and 57. The results are reported in Table 3.4 and clearly show that TabuMDP with $t = 1$ is again not competitive at all on these graphs. Indeed, the deficiency obtained by using only N_1 increases linearly with the number of vertices, while Proposition 6 demonstrates

Table 3.4: Results for Schwartz's graphs

In-	$\|V\|$	$\|E\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			$Def(G)$
stance			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
$S(25)$	626	7825	24*	45	34835	28	88	13643880	260	30	12596676	24
$S(33)$	1090	17985	32*	56	76017	39	102	6942400	420	40	4454546	32
$S(41)$	1682	34481	40*	72	218171	43	125	3255052	621	48	2568343	40
$S(49)$	2402	58849	49	82	1109484	52	139	1151476	926	56	1188054	48
$S(57)$	3250	92625	61	97	663103	63	141	674954	1246	65	734938	56

that the optimal increase should be proportional to the square root of the number of vertices (since $S(k)$ has $k^2 + 1$ vertices while $Def(S(k)) = k - 1$). Notice that Proposition 6 also demonstrates that at least $\frac{3k-1}{2}$ colors are needed to reach an optimal deficiency. The use of N_1 in TabuMDP is therefore probably very helpful to increase the number of colors from the initial value $\Delta(G) + 1 = k + 1$ to $\frac{3k-1}{2}$.

We also observe that, in this case, TabuMDP with $t = 100$ performs better than TabuMDP*. This is probably due to the fact that TabuMDP with $t = 100$ easily generates edge colorings c of $S(k)$ with $Span(c) \geq \frac{3k-1}{2}$, while the bonus in TabuMDP* is so effective that it increases the span to much higher values, and this induces an increase in deficiency.

Note that we have solved the MDP to optimality for Schwartz's graphs having up to 1682 vertices. By increasing the computing time to one hour, an optimal solution can also be found for $S(49)$ which contains 2402 vertices.

3.4.5 Complete bipartite graphs

A graph is bipartite if its vertex set can be divided into two disjoint sets A and B such that no edge has both endpoints in the same set. It is complete bipartite if every vertex in A is linked to every vertex in B. We denote $K(p, q)$ the complete

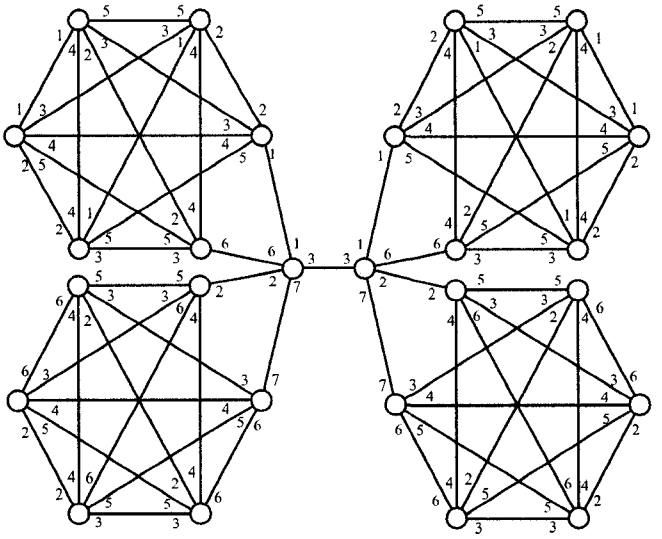


Figure 3.6: An edge coloring c of $S(5)$ with $D(S(5), c) = 4$.

bipartite graph with $|A| = p$ and $|B| = q$. Complete bipartite graphs are interesting instances to be tested since they admit consecutive colorings (i.e., edge colorings with no deficiency), while the number of colors needed to reach such an optimal deficiency can be very large. More precisely, for a complete bipartite graph $K(p, q)$, it is known [8] that an edge coloring c with $D(G, c) = 0$ uses at least $p + q - gcd(p, q)$ colors, where $gcd(p, q)$ is the greatest common divisor of p and q.

We have performed tests on complete bipartite graphs with pairs (p, q) equal to (5,7), (11,13), (17,19), (20,25), and (29, 31). As can be observed from the results in Table 3.5, even though we have chosen instances with a very small number of vertices, the only optimal solution that could be produced is for a graph with 12 vertices. With more than 100 million iterations, we have not been able to produce any consecutive coloring of $K(11, 13)$ which contains only 24 vertices and 143 edges. This probably means that a consecutive coloring of a complete bipartite graph can only be obtained by using neighborhoods that differ from N_1 and N_2. Note that, for any strictly positive even number k, the complete bipartite graph $K(k - 1, k + 1)$

Table 3.5: Results for complete bipartite graphs

Instance	$\|V\|$	$\|E\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			$Def(G)$
			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
$K(5,7)$	12	35	0*	11	35657	0*	11	260	0*	11	509311	0
$K(11,13)$	24	143	4	20	183753845	6	22	166710084	8	19	126745367	0
$K(17,19)$	36	323	8	30	144342410	5	33	123728982	20	25	71280150	0
$K(20,25)$	45	500	15	39	81477699	11	42	65686936	55	33	43777641	0
$K(29,31)$	60	899	24	46	98520574	10	54	60715001	44	37	31531508	0

has maximum degree $\Delta(K(k-1,k+1)) = k+1$ while it follows from [8] that $(k-1)+(k+1)-gcd(k-1,k+1) = 2k-1$ colors are needed to color the edges of $K(k-1,k+1)$ without any deficiency. This very large number of colors when compared to the maximum degree probably explains the poor performance of our algorithms.

3.4.6 Hertz's graphs

Hertz [10] has shown through a simple example that bipartite graphs can have a strictly positive deficiency. Giaro et al. [4] have generalized this example to what they call the *Hertz's graphs*. These graphs, defined for every two integers $p \geq 4$ and $q \geq 3$, are denoted $H(p,q)$ and constructed as follows. First, create a vertex a adjacent to p vertices $b_1, \ldots, b_p$. Then connect each b_i to q vertices $c_{i,1}, \ldots, c_{i,q}$. Finally, connect every $c_{i,j}$ to a vertex d. For illustration, the graph $H(4,3)$ is represented in Figure 3.7.

A graph $H(p,q)$ has $pq+p+2$ vertices. It is proved in [4] that $Def(H(p,q)) = pq-p-2q-2$, and that there exists an optimal edge coloring c of $H(p,q)$ that uses pq colors. We have performed tests on Hertz's graphs with $p = q$. Note that $lim_{p\to\infty} \frac{Def(H(p,p))}{|V(H(p,p))|} = 1$, which means that the deficiency increases linearly with the

number of vertices in $H(p,p)$. Results obtained on Hertz's graphs $H(p,p)$ with p =20, 25, 30, and 35 are reported in Table 3.6.

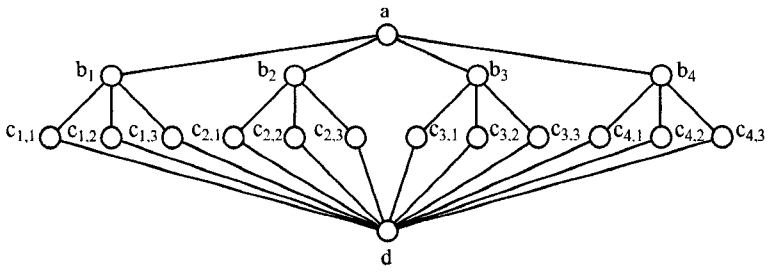


Figure 3.7: The Hertz's graph $H(4,3)$.

TabuMDP produces surprisingly good solutions with parameter $t = 1$ and bad ones with $t = 100$. While both versions of TabuMDP produce edge colorings of $H(p,p)$ with the same number p^2 of colors, the computed deficiency is much higher with $t = 100$. In fact, the use of neighborhood N_1 makes it possible to reach an optimal edge coloring for Hertz's graphs having up to 1262 vertices, while the use of neighborhood N_2 appears as particularly ineffective on these graphs. We have tried to find an explanation to such a behavior. One reason might be the range of values of the degrees of the vertices in $H(p,p)$ which vary from 2 (there are p^2 such vertices) to p^2 (there is a unique vertex with such a high degree), while all vertices had almost the same degree in the previous tests. In our experiments, we have also observed that $N_2(c)$ typically contains many solutions with the same deficiency as $D(G,c)$, while this is not the case for $N_1(c)$. Hence, in order to escape from a local optimum while using N_2, the tabu list has to be much larger. Tests with a tabu list of size $\frac{|E|}{5}$ (instead of $\sqrt{|E|}$) have indeed made it possible to get better results, but still not as good as those produced while using N_1 only.

Table 3.6: Results for Hertz's graphs

In-	$\|V\|$	$\|E\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			$Def(G)$
stance			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
$H(20,20)$	422	820	432	400	13994038	1241	415	9708321	338*	400	2658	338
$H(25,25)$	652	1275	1902	625	4365111	1600	626	6334557	548*	625	4564	548
$H(30,30)$	932	1830	2656	900	2849465	3710	1077	3012199	808*	900	7292	808
$H(35,35)$	1262	2485	5895	1225	1430847	11789	1352	565982	1118*	1225	11235	1118

3.4.7 Extended hypercubic graphs

For a strictly positive integer k, the *hypercubic graph* $HC(k)$ is the graph with vertex set V equal to the set of all possible binary strings of length k, and where two vertices are linked by an edge if and only if the two corresponding strings differ in precisely one bit. Hence, $HC(k)$ is connected, has 2^k vertices, is k-regular, and $Diam(HC(k)) = k$.

Let $HC^*(k)$, called *extended hypercubic graph*, be the graph obtained from $HC(k)$ by adding a vertex v adjacent to all vertices of $HC(k)$. It follows from Corollary 3 that $Def(HC^*(k)) \geq (2^k + 1) - k^2 - k - 2 = 2^k - k^2 - k - 1$, which means that $lim_{k\to\infty} \frac{Def(HC^*(k))}{|V(HC^*(k))|} \geq 1$. For illustration, the extended hypercubic graph $HC^*(2)$ is represented in Figure 3.8.

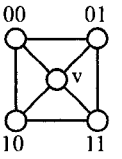


Figure 3.8: The extended hypercubic graph $HC^*(2)$.

Results on extended hypercubic graphs $HC^*(k)$ with $k = 4$, 5, 6, and 7 are reported in Table 3.7. The last column contains the proved lower bound $L = \max\{0, 2^k - k^2 - k - 1\}$ on $Def(HC^*(k))$. While this lower bound grows linearly

Table 3.7: Results for extended hypercubic graphs

Instance	$\|V\|$	$\|E\|$	**TabuMDP** $t = 100$			**TabuMDP*** $t = 100$			**TabuMDP** $t = 1$			L
			$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	$D(G,c)$	$Span(c)$	Iterations	
$HC^*(4)$	17	48	7	16	285960857	7	16	314895161	11	16	206188926	0
$HC^*(5)$	33	112	80	32	77188984	80	32	129520880	105	32	120087299	1
$HC^*(6)$	65	256	477	64	29559250	464	64	34561557	663	64	38442527	21
$HC^*(7)$	129	576	2368	128	5752621	2434	128	6492483	3228	128	9062580	71

with the number of vertices, our algorithms seem to produce edge colorings with a quadratic increase of $D(G, c)$. This may be due to a poor lower bound L or to a poor performance of our algorithms.

There is no clear winner between TabuMDP and TabuMDP* (with $t = 100$). However, the use of neighborhood N_2 clearly helps reducing the deficiency since the results are better with $t = 100$ than with $t = 1$.

3.5 Final remarks and conclusion

We have developed a tabu search algorithm for the minimum deficiency problem. It uses two types of neighborhoods, one very simple, where the color of exactly one edge is modified, and a more complex one based on bichromatic exchanges. The results of the experiments reported in Section 3.4 clearly indicate that the use of the second neighborhood N_2 helps, in general, to reduce the deficiency. One exception occurred for Hertz's graphs for which it seems more difficult to escape from a local optimum when using N_2 instead of N_1.

The regular use of N_1 is often crucial to help the algorithms increase the number of colors. This is particularly important for graphs with an odd number of vertices which often require much more colors than $\Delta(G) + 1$ to reach the optimal deficiency.

When the use of N_1 is not sufficient to increase the number of colors, we have seen that the variant TabuMDP* can help in this task. The superiority of TabuMDP* over TabuMDP has been demonstrated for random graphs, random k-regular graphs, and complete graphs with an odd number of vertices. TabuMDP* is however not competitive with TabuMDP for graphs with an even number of vertices.

The reported experiments clearly demonstrate that graphs with an odd number of vertices are more challenging for the MDP. Indeed, one of the main difficulties of the MDP is to determine the number of colors that are required to obtain an optimal deficiency. We always start with an edge coloring that uses $\Delta(G)$ or $\Delta(G)+1$ colors. This number must sometimes be raised to a much higher value, especially for graphs with an odd number of vertices. Note however that Schwartz's graphs are k-regular graphs with an even number of vertices while optimal edge colorings use at least $\frac{3k-1}{2}$ colors. An increase from $k+1$ to $\frac{3k-1}{2}$ is therefore also required in this case.

Within 20 minutes, we have been able to solve to optimality instances having up to 1682 vertices. On the other hand, complete bipartite graphs constitute an exception since they all admit consecutive colorings (i.e., edge colorings c with $D(G,c) = 0$) while no such edge colorings could be produced by our algorithms for such graphs with 24 vertices and more.

Finally, note that extended hypercubic graphs seem to have very high deficiencies. While the proved lower bound on $Def(HC^*(k))$ is strictly smaller than the number of vertices in $HC^*(k)$, TabuMDP has produced much larger deficiencies. To date, no graphs $G = (V,E)$ are known with $Def(G) > |V|$. It would therefore be interesting to determine if extended hypercubic graphs satisfy this property.

BIBLIOGRAPHY

[1] Asratian, A.S., Casselgren, C.J. (2006). On interval edge colorings of (α, β)-biregular bipartite graphs. *Discrete Mathematics* 307:1951–1956.

[2] Asratian, A.S., Kamalian, R.R. (1987). Interval colorings of the edges of a multigraph. *Applied Math.* 5:25–34, (in Russian).

[3] Giaro, K. (1997). The complexity of consecutive Δ-coloring of bipartite graphs: 4 is easy, 5 is hard. *Ars Combinatoria* 47:287–298.

[4] Giaro, K., Kubale, M., Malafiejski, M. (1999). On the deficiency of bipartite graphs. *Discrete Applied Mathematics* 94:193–203, 1999.

[5] Giaro, K., Kubale, M., Malafiejski, M. (1999). Compact scheduling in open shop with zero-one time operations. *INFOR* 37:37–47.

[6] Giaro, K., Kubale, M., Malafiejski, M. (2001). Consecutive colorings of the edges of general graphs. *Discrete Mathematics* 236:131–143, 2001.

[7] Glover, F., Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, Boston.

[8] Hanson, D., Loten, C.O.M., Toft, B. (1996). A lower bound for interval colouring of bi-regular bipartite graphs. *Bulletin of the Institute of Combinatorics and Applications* 18:69–74.

[9] Hanson, D., Loten, C.O.M., Toft, B. (1998). On interval colorings of bi-regular bipartite graphs. *Ars Combinatoria* 50:23–32.

[10] Hertz, A. (1995). Email information.

[11] Kim, J.H., Vu, V.H. (2003). Generating random regular graphs. *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC '03)*, San Diego, USA. ACM Press, New York. 213-222.

[12] Piatkin, A.V. (2004). Interval coloring of $(3, 4)$-biregular bipartite graphs having large cubic subgraphs. *Journal of Graph Theory* 47(2):122–128.

[13] Schwartz, A. (2006). The deficiency of a regular graph. *Discrete Mathematics* 306:1947–1954.

[14] Sevastianov, S.V. (1990). On interval edge colouring of bipartite graphs *Metody Diskret. Analiz.* 50:61–72, (in Russian).

[15] Vizing, V.G. (1964). On an estimate of the chromatic class of a p-graph. *Discret. Analyz.* 3:25–30.

CHAPITRE 4 : ON A REDUCTION OF THE INTERVAL COLORING PROBLEM TO A SERIES OF BANDWIDTH COLORING PROBLEMS

Soumis dans *Journal of Scheduling* le 3 avril 2008.

On a reduction of the interval coloring problem to a series of bandwidth coloring problems

Mathieu Bouchard [1] **Mirjana Čangalović** [2] **Alain Hertz** [1]

[1] *École Polytechnique and GERAD, Montréal, Canada*

{Mathieu.Bouchard, Alain.Hertz}@gerad.ca

[2] *Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia*

canga@fon.bg.ac.yu

Abstract

Given a graph $G = (V, E)$ with strictly positive integer weights ω_i on the vertices $i \in V$, an interval coloring of G is a function I that assigns an interval $I(i)$ of ω_i consecutive integers (called colors) to each vertex $i \in V$ so that $I(i) \cap I(j) = \emptyset$ for all edges $\{i, j\} \in E$. The interval coloring problem is to determine an interval coloring that uses as few colors as possible. Assuming that a strictly positive integer weight δ_{ij} is associated with each edge $\{i, j\} \in E$, a bandwidth coloring of G is a function c that assigns an integer (called a color) to each vertex $i \in V$ so that $|c(i) - c(j)| \geq \delta_{ij}$ for all edges $\{i, j\} \in E$. The bandwidth coloring problem is to determine a bandwidth coloring with minimum difference between the largest and the smallest colors used. We prove that an optimal solution of the interval coloring problem can be obtained by solving a series of bandwidth coloring problems. Computational experiments demonstrate that such a reduction can help to solve larger instances or to obtain better upper bounds on the optimal solution value of the interval coloring problem.

4.1 Introduction

Given a graph $G = (V, E)$ with vertex set V and edge set E, the graph coloring problem is to assign a color to each vertex so that no two adjacent vertices have the same color and the total number of different colors is minimized. We consider two generalizations of this problem, namely the bandwidth and the interval coloring problems.

Assuming that a strictly positive integer weight δ_{ij} is associated with each edge $\{i, j\} \in E$, a bandwidth coloring of G is a function c that assigns an integer (called a color) to each vertex $i \in V$ so that $|c(i) - c(j)| \geq \delta_{ij}$ for all edges $\{i, j\} \in E$. Denoting $max(c) = \max_{i \in V} c(i)$ and $min(c) = \min_{i \in V} c(i)$ the largest and smallest colors used in c, the *span* of a bandwidth coloring c is defined as $span(c) = max(c) - min(c) + 1$. The bandwidth coloring problem is to determine a bandwidth coloring with minimum span. The graph coloring problem is a special case of the bandwidth coloring problem with $\delta_{ij} = 1$ for all edges $\{i, j\} \in E$.

The second generalization of the graph coloring problem assumes that a strictly positive integer weight ω_i is associated with each vertex $i \in V$. An interval coloring of G is a function I that assigns an interval $I(i)$ of ω_i consecutive integers (called colors) to each vertex $i \in V$ so that $I(i) \cap I(j) = \emptyset$ for all edges $\{i, j\} \in E$. The interval coloring problem is to determine an interval coloring that uses as few colors as possible. The special case with $\omega_i = 1$ for all vertices $i \in V$ is equivalent to the graph coloring problem. The interval coloring problem has a fairly long history dating back, at least to the 1970s. For example, Stockmeyer showed in 1976 that the interval-coloring problem is NP-hard, even when restricted to interval graphs and vertex weights in $\{1, 2\}$ (see problem SR2 in [6]). In 1976, Punter has formulated and solved a school timetabling problem with non-preemptive multiple period lessons using an interval coloring model. Another early application of the interval coloring problem was in the compile-time memory-allocation problem [5].

The circular coloring problem of weighted graphs, introduced by Deuber and Zhu [4], has similarities with the interval coloring problem. More precisely, let p be

a strictly positive integer, and define a p-circular coloring of G as a function I that assigns to each vertex $i \in V$ an interval $I(i)$ of ω_i consecutive integers (called colors) in $\{0, \cdots, p-1\}$, where 0 is considered as consecutive to $p-1$, and $I(i) \cap I(j) = \emptyset$ for all edges $\{i, j\} \in E$. The circular coloring problem is to determine the smallest integer p, denoted $\chi_c(G)$, such that there exists a p-circular coloring of G. By denoting $\chi_{int}(G)$ the optimal value of the interval coloring problem, we have $\chi_c(G) \leq \chi_{int}(G)$ for all graphs G since every interval coloring of G that uses p colors is a p-circular coloring of G. It may happen that $\chi_c(G)$ is strictly smaller than $\chi_{int}(G)$. For example, if G is a cycle on five vertices and all vertex weights equal 2, then $\chi_c(G) = 5$ while $\chi_{int}(G) = 6$. It is not difficult to prove that $\chi_c(G) \leq \chi_{int}(G) < \chi_c(G) + \max_{i \in V} w_i$.

Since the graph coloring problem is NP-hard [6], both the bandwidth and the interval coloring problems are NP-hard too. In 2002, Prestwich [7] has described an exact algorithm for solving the bandwidth coloring problem, while an exact algorithm for the interval coloring problem was proposed by Čangalović and Schreuder [2] in 1991.

The aim of this paper is to show that an optimal solution of the interval coloring problem can be obtained by solving a series of bandwidth coloring problems. This reduction is described in the next section with a proof of its validness. Section 4.3 will be devoted to computational experiments, where we compare the CPU time needed to solve the interval coloring problem using the algorithm in [2] with the total CPU time needed to solve the series of bandwidth coloring problems using the algorithm in [7].

4.2 The proposed reduction

4.2.1 Preliminary observations

The main idea of our reduction is based on the simple observation that if two intervals of length ℓ_1 and ℓ_2 are not intersecting, then the distance between their centers is at

least $\frac{\ell_1+\ell_2}{2}$. Proposition 7 uses this relation to show the one-to-one correspondence between the bandwidth colorings and the interval colorings of G when all vertex weights are even.

Proposition 7. *Let $G = (V, E)$ be a graph with even weights ω_i on the vertices $i \in V$ and with weights $\delta_{ij} = \frac{\omega_i+\omega_j}{2}$ on the edges $\{i, j\} \in E$. Then c is a bandwidth coloring of G if and only if the intervals $I(i) = \{c(i) - \frac{\omega_i}{2}, \cdots, c(i) + \frac{\omega_i}{2} - 1\}$ define an interval coloring of G.*

proof. Let c be a bandwidth coloring of G. The intervals $I(i) = \{c(i) - \frac{\omega_i}{2}, \cdots, c(i) + \frac{\omega_i}{2} - 1\}$ contain exactly ω_i consecutive integers. To prove that they define an interval coloring of G, we show that $I(i) \cap I(j) = \emptyset$ for all edges $\{i, j\} \in E$. So consider any edge $\{i, j\} \in E$, and assume, without loss of generality, that $c(i) < c(j)$. Then $c(j) \geq c(i) + \frac{\omega_i+\omega_j}{2}$, and the intervals $I(i)$ and $I(j)$ do not intersect since

$$c(j) - \frac{\omega_j}{2} \geq c(i) + \frac{\omega_i + \omega_j}{2} - \frac{\omega_j}{2} = c(i) + \frac{\omega_i}{2}.$$

Conversely, let I be an interval coloring of G and let $min_I(i)$ denote the smallest color in an interval $I(i)$. Consider any edge $\{i, j\} \in E$, and assume, without loss of generality, that $min_I(i) < min_I(j)$. Then $min_I(j) \geq min_I(i) + \omega_i$, which means that

$$c(j) - c(i) = min_I(j) + \frac{\omega_j}{2} - min_I(i) - \frac{\omega_i}{2} \geq \frac{\omega_j + \omega_i}{2} = \delta_{ij}.$$

Hence, c is a bandwidth coloring of G. □

Given a graph $G = (V, E)$ with even weights ω_i on the vertices $i \in V$ and with weights $\delta_{ij} = \frac{\omega_i + \omega_j}{2}$ on the edges $\{i, j\} \in E$, Proposition 7 demonstrates that there is a one-to-one correspondence between the bandwidth and the interval colorings of G. Notice however that an optimal bandwidth coloring of G does not necessarily correspond to an optimal interval coloring, and vice versa. Consider for example, the graph in Fig. 4.1(a), where the numbers into boxes correspond to weights. An optimal bandwidth coloring c of G with $span(c) = 5$ is represented in Fig. 1(b). The corresponding interval coloring shown in Fig. 1(c) uses 9 colors, which is not optimal. An optimal interval coloring of G with 8 different colors is represented in Fig. 4.1(d). The corresponding bandwidth coloring has span 6, as shown in Fig. 4.1(e).

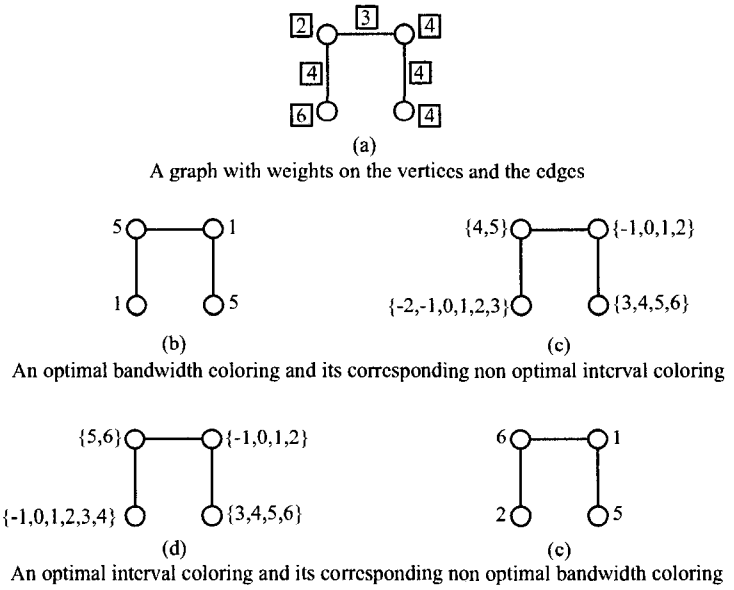


Figure 4.1: Non correspondence between optimal bandwidth and interval colorings.

If the weights on the vertices are not all even, then the intervals in Proposition 7 contain non integer values, and the weights δ_{ij} on the edges are possibly non integer. We will show in Section 4.2.2.2 how to handle graphs having possibly odd vertex weights.

4.2.2 The algorithm and its validness

In this section, we describe a procedure that determines an optimal interval coloring by solving a series of bandwidth coloring problems. We first consider graphs with even vertex weights, and we then extend the approach to graphs with general vertex weights. In what follows, we say that an interval coloring I is *compact* if it uses all colors in $\{\min_{i\in V} min_I(i), \cdots, \max_{i\in V}(min_I(i)+\omega_i)-1\}$.

4.2.2.1 Graphs with even vertex weights

Let $G=(V,E)$ be a graph with even weights ω_i on the vertices $i \in V$, and let Δ be any positive integer. We denote G_Δ the edge-weighted graph obtained from G by adding two adjacent vertices α and β linked to all vertices in V, and by setting:

- $\delta_{ij} = \frac{\omega_i+\omega_j}{2}$ for all edges $\{i,j\} \in E$,
- $\delta_{\alpha i} = \delta_{\beta i} = \frac{\omega_i}{2}$ for all vertices $i \in V$,
- $\delta_{\alpha\beta} = \Delta$.

Let $\chi_{int}(G)$ and $\chi_b(G)$ denote the optimal values of the interval and the bandwidth coloring problems on a graph G.

Proposition 8. *If* $\Delta \leq \chi_{int}(G)$, *then* $\chi_{int}(G) \geq \chi_b(G_\Delta) - 1$.

proof. Consider the graph G_Δ for an integer $\Delta \leq \chi_{int}(G)$, and let I be a compact optimal interval coloring of G. Define the colors $c_\Delta(i)$ on the vertices of G_Δ as follows:

- $c_\Delta(i) = min_I(i) + \frac{\omega_i}{2}$ for all $i \in V$,
- $c_\Delta(\alpha) = \min_{i\in V} min_I(i)$,
- $c_\Delta(\beta) = \max_{i\in V}(min_I(i)+\omega_i)$.

Since I uses $\max_{i \in V}(min_I(i) + \omega_i) - \min_{i \in V} min_I(i)$ different colors, we have $c_\Delta(\beta) - c_\Delta(\alpha) = \chi_{int}(G)$. Moreover, the colors $c_\Delta(i)$ define a bandwidth coloring of G_Δ. Indeed,

- $|c_\Delta(j) - c_\Delta(i)| \geq \delta_{ij}$ for all edges $\{i, j\} \in E$, as shown in Proposition 7,
- $|c_\Delta(i) - c_\Delta(\alpha)| = min_I(i) + \frac{\omega_i}{2} - \min_{j \in V} min_I(j) \geq \frac{\omega_i}{2} = \delta_{\alpha i}\ \forall i \in V$,
- $|c_\Delta(\beta) - c_\Delta(i)| = \max_{j \in V}(min_I(j) + \omega_j) - min_I(i) - \frac{\omega_i}{2} \geq \frac{\omega_i}{2} = \delta_{\beta i}\ \forall i \in V$,
- $|c_\Delta(\beta) - c_\Delta(\alpha)| = \chi_{int}(G) \geq \Delta = \delta_{\alpha\beta}$.

Since $\chi_{int}(G) = c_\Delta(\beta) - c_\Delta(\alpha) = span(c_\Delta) - 1$, we conclude that $\chi_{int}(G) \geq \chi_b(G_\Delta) - 1$. □

Corollary 5. *Let c_Δ be an optimal bandwidth coloring of G_Δ. If $\Delta \leq \chi_{int}(G)$ and $|c_\Delta(\beta) - c_\Delta(\alpha)| = span(c_\Delta) - 1$ then $\chi_{int}(G) = \chi_b(G_\Delta) - 1$ and the intervals $I(i) = \{c_\Delta(i) - \frac{\omega_i}{2}, \cdots, c_\Delta(i) + \frac{\omega_i}{2} - 1\}$ define a compact optimal interval coloring of G.*

proof. Without loss of generality, assume $c_\Delta(\alpha) \leq c_\Delta(\beta)$. Since $span(c_\Delta) = c_\Delta(\beta) - c_\Delta(\alpha) + 1$, we have $c_\Delta(\alpha) \leq c_\Delta(i) - \frac{\omega_i}{2}$ and $c_\Delta(\beta) \geq c_\Delta(i) + \frac{\omega_i}{2}$ for all vertices $i \in V$. As shown in Proposition 7, the intervals $I(i) = \{c_\Delta(i) - \frac{\omega_i}{2}, \cdots, c_\Delta(i) + \frac{\omega_i}{2} - 1\}$ define an interval coloring of G. Such a coloring uses at most $\max_{i \in V}(min_I(i) + \omega_i) -$

$\min_{i \in V} min_I(i) \leq c_\Delta(\beta) - c_\Delta(\alpha)$ different colors, which means that

$$\begin{aligned} \chi_{int}(G) &\leq c_\Delta(\beta) - c_\Delta(\alpha) \\ &= span(c_\Delta) - 1 = \chi_b(G_\Delta) - 1 \\ &\leq \chi_{int}(G) \text{ (from Proposition 8).} \end{aligned}$$

Hence $\chi_{int}(G) = \chi_b(G_\Delta) - 1$ and the interval coloring I is optimal. The above inequalities also imply that $\chi_{int}(G) = \max_{i \in V}(min_I(i) + \omega_i) - \min_{i \in V} min_I(i)$, which means that I is compact. □

The proposed algorithm for determining an optimal interval coloring of a graph G with even vertex weights by solving a series of bandwidth coloring problems is described in Fig. 4.2. It consists in determining optimal bandwidth colorings c_{Δ_k} in graphs G_{Δ_k} for various values of Δ_k, until $|c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| = span(c_{\Delta_k}) - 1$.

Theorem 4.2.1. *The EvenReduction algorithm is finite and the intervals $I(i)$ produced as output define a compact optimal interval coloring of G*

proof. We first prove that $\Delta_k \leq \chi_{int}(G)$ at each iteration k. This is obviously true for $k = 1$ since $\Delta_1 = 0 < \chi_{int}(G)$. So assume $\Delta_k \leq \chi_{int}(G)$ and $|c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| < span(c_{\Delta_k}) - 1$. Then

$$\Delta_{k+1} = span(c_{\Delta_k}) - 1 = \chi_b(G_{\Delta_k}) - 1 \leq \chi_{int}(G)$$

Algorithm EvenReduction

Input A graph $G = (V, E)$ with even weights ω_i on the vertices $i \in V$;

Output An optimal interval coloring I of G;

(1) Set $\Delta_1 \leftarrow 0$ and $k \leftarrow 1$;

(2) Determine an optimal bandwidth coloring c_{Δ_k} of G_{Δ_k};

Set $\Delta_{k+1} \leftarrow span(c_{\Delta_k}) - 1$;

(3) If $|c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| < \Delta_{k+1}$ then set $k \leftarrow k + 1$, and go to (2);

Else set $I(i) = \{c_{\Delta_k}(i) - \frac{\omega_i}{2}, \cdots, c_{\Delta_k}(i) + \frac{\omega_i}{2} - 1\}$ for all $i \in V$ and STOP.

Figure 4.2: The proposed algorithm for graphs with even vertex weights.

the last inequality being valid according to Proposition 8. Now, since

$$\Delta_{k+1} = span(c_{\Delta_k}) - 1 > |c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| \geq \Delta_k$$

we know that the algorithm is finite. Finally, since the algorithm stops with $|c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| = span(c_{\Delta_k}) - 1$, it follows from Corollary 5 that the intervals $I(i) = \{c_{\Delta_k}(i) - \frac{\omega_i}{2}, \cdots, c_{\Delta_k}(i) + \frac{\omega_i}{2} - 1\}$ define a compact optimal interval coloring of G. □

Let k^* denote the final value of the iteration counter in the EvenReduction algorithm. It corresponds to the number of bandwidth coloring problems which have to be solved to determine an optimal interval coloring of G. The following result is a direct consequence of the proof of Theorem 4.2.1.

Corollary 6. $\Delta_1 < \cdots < \Delta_{k^*} \leq \chi_{int}(G)$.

4.2.2.2 Graphs with general vertex weights

The EvenReduction procedure cannot be applied to graphs with possibly odd vertex weights. Indeed, if ω_i is odd, then the extreme values $c_{\Delta_k}(i) - \frac{\omega_i}{2}$ and $c_{\Delta_k}(i) + \frac{\omega_i}{2} - 1$ of the intervals $I(i)$ are not integer. Also, if an edge $\{i,j\} \in E$ links two vertices with weights of different parity, then the weight $\delta_{ij} = \frac{\omega_i + \omega_j}{2}$ of $\{i,j\}$ in G_{Δ_k} is not integer. The next Proposition shows how to obtain an optimal interval coloring for graphs with general vertex weights.

Proposition 9. *Let $G = (V,E)$ be a graph with weights ω_i on the vertices $i \in V$, and let G' be the same graph as G except that the vertices have weight $\omega'_i = 2\omega_i$ in G'. Given any compact optimal interval coloring I' of G', the intervals $I(i) = \{\left\lceil \frac{min_{I'}(i)}{2} \right\rceil, \cdots, \left\lceil \frac{min_{I'}(i)}{2} \right\rceil + \omega_i - 1\}$ define a compact optimal interval coloring of G.*

proof. Since all weights in G' are even, it is proved in [1] that $\chi_{int}(G') = 2\chi_{int}(G)$. Let I' be any compact optimal interval coloring of G' and let u be a vertex with minimum value $min_{I'}(u)$ and v a vertex with maximum value $min_{I'}(v) + \omega'_v$. Then $\chi_{int}(G') = min_{I'}(v) + \omega'_v - min_{I'}(u)$, and since $\chi_{int}(G')$ is even, we know that $min_{I'}(u)$ and $min_{I'}(v)$ have the same parity. Hence,

$$\begin{aligned} \chi_{int}(G) &= \frac{\chi_{int}(G')}{2} \\ &= \frac{min_{I'}(v) + \omega'_v - min_{I'}(u)}{2} \\ &= \left\lceil \frac{min_{I'}(v)}{2} \right\rceil - \left\lceil \frac{min_{I'}(u)}{2} \right\rceil + \omega_v \end{aligned}$$

Define the intervals $I(i) = \{\left\lceil \frac{min_{I'}(i)}{2} \right\rceil, \cdots, \left\lceil \frac{min_{I'}(i)}{2} \right\rceil + \omega_i - 1\}$ for every vertex $i \in V$.

Since the smallest integer in $\cup_{i \in V} I(i)$ is $\left\lceil \frac{min_{I'}(u)}{2} \right\rceil$ while the largest is $\left\lceil \frac{min_{I'}(v)}{2} \right\rceil + \omega_v - 1$, these intervals use $\left\lceil \frac{min_{I'}(v)}{2} \right\rceil + \omega_v - \left\lceil \frac{min_{I'}(u)}{2} \right\rceil = \chi_{int}(G)$ different colors.

It remains to prove that I is an interval coloring of G. Consider any edge $\{i, j\} \in E$. Since the intervals $I'(i)$ and $I'(j)$ do not intersect, assume without loss of generality that $min_{I'}(i) \geq min_{I'}(j) + \omega'_j$.

- If $min_{I'}(i)$ and $min_{I'}(j)$ have the same parity, then

$$min_I(i) - min_I(j) = \frac{min_{I'}(i) - min_{I'}(j)}{2} \geq \frac{\omega'_j}{2} = \omega_j.$$

- if $min_{I'}(i)$ and $min_{I'}(j)$ have different parity, then $min_{I'}(i) - min_{I'}(j) \geq \omega'_j + 1$, which means that

$$min_I(i) - min_I(j) \geq \frac{min_{I'}(i) - min_{I'}(j) - 1}{2} \geq \frac{\omega'_j}{2} = \omega_j.$$

Hence, $I(i)$ and $I(j)$ do not intersect. □

The above proof together with Theorem 4.2.1 demonstrate that the following algorithm determines an optimal interval coloring I of any graph G.

The algorithm is illustrated on Fig. 4.3, where the numbers into boxes correspond to weights.

Algorithm GeneralReduction

Input A graph $G = (V, E)$ with weights ω_i on the vertices $i \in V$;

Output An optimal interval coloring I of G;

(1) Construct G' from G by multiplying every weight ω_i by 2;

(2) Determine a compact optimal interval coloring I' of G' using EvenReduction;

(3) Set $I(i) = \{\left\lceil \frac{min_{I'}(i)}{2} \right\rceil, \cdots, \left\lceil \frac{min_{I'}(i)}{2} \right\rceil + \omega_i - 1\}$ for all vertices $i \in V$.

Fig. 4.2.2.2. The proposed algorithm for graphs with general vertex weights.

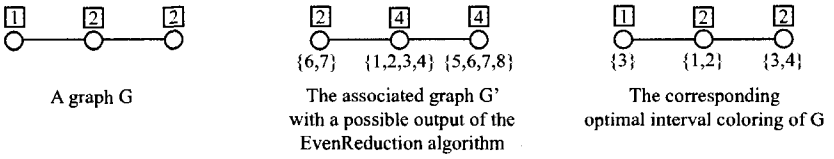


Figure 4.3: Illustration of the GeneralReduction algorithm.

4.2.3 Complexity analysis

Let $G = (V, E)$ be a graph with even weights ω_i on the vertices $i \in V$, and let k^* denote the number of bandwidth coloring problems which have to be solved when applying EvenReduction on G. In order to determine an upper bound on k^*, we now associate a weight $\omega_\alpha = \omega_\beta = 0$ to vertices α and β in G_{Δ_k}, and we always assume, without loss of generality, that $c_{\Delta_k}(\alpha) \leq c_{\Delta_k}(\beta)$. For a function f on a set S, let $argmin_{x \in S} f(x)$ and $argmin_{x \in S} f(x)$ respectively denote the set of elements $x \in S$ with minimum and maximum value $f(x)$. For every iteration k of the algorithm, we define

$$\ell_k = \begin{cases} \alpha & \text{if } c_{\Delta_k}(\alpha) < \min_{i \in V} c_{\Delta_k}(i) \\ \text{any vertex in } argmin_{i \in V}(c_{\Delta_k}(i) - \frac{\omega_i}{2}) & \text{otherwise} \end{cases}$$

$$u_k = \begin{cases} \beta & \text{if } c_{\Delta_k}(\beta) > \max_{i \in V} c_{\Delta_k}(i) \\ \text{any vertex in } argmax_{i \in V}(c_{\Delta_k}(i) + \frac{\omega_i}{2}) & \text{otherwise} \end{cases}$$

Proposition 10. *For every iteration $k < k^*$, we have*

(a) $\Delta_{k+1} - \Delta_k \geq \frac{\omega_{\ell_k} + \omega_{u_k}}{2} \geq \chi_{int}(G) - \Delta_{k+1}$,

(b) $\chi_{int}(G) - \Delta_k \geq 2(\chi_{int}(G) - \Delta_{k+1})$,

(c) If $k \geq 3$ *then* $\omega_{\ell_{k-2}} + \omega_{u_{k-2}} > \omega_{\ell_{k-1}} + \omega_{u_{k-1}}$.

proof. Notice first that

$$c_{\Delta_k}(\alpha) \geq c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2}.$$

Indeed, this is true for $\ell_k = \alpha$ since $c_{\Delta_k}(\alpha) = c_{\Delta_k}(\alpha) + \frac{\omega_\alpha}{2}$. For $\ell_k \neq \alpha$, we have $c_{\Delta_k}(\ell_k) < c_{\Delta_k}(\alpha)$, which means that $c_{\Delta_k}(\alpha) - c_{\Delta_k}(\ell_k) \geq \delta_{\alpha\ell_k} = \frac{\omega_{\ell_k}}{2}$. Similarly,

$$c_{\Delta_k}(\beta) \leq c_{\Delta_k}(u_k) - \frac{\omega_{u_k}}{2}.$$

We therefore have

$$\begin{aligned} \Delta_k &\leq c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha) \\ &\leq c_{\Delta_k}(u_k) - \frac{\omega_{u_k}}{2} - c_{\Delta_k}(\ell_k) - \frac{\omega_{\ell_k}}{2}. \end{aligned} \tag{4.2.1}$$

Also, it follows from the definitions of Δ_{k+1} and $span(c_{\Delta_k})$ that

$$\Delta_{k+1} = span(c_{\Delta_k}) - 1 \geq c_{\Delta_k}(u_k) - c_{\Delta_k}(\ell_k). \tag{4.2.2}$$

From (4.2.1) and (4.2.2), we deduce the left inequality in (a) since

$$\begin{aligned}\Delta_{k+1} - \Delta_k &\geq c_{\Delta_k}(u_k) - c_{\Delta_k}(\ell_k) - c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} + c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2} \\ &= \frac{\omega_{\ell_k} + \omega_{u_k}}{2}.\end{aligned}$$

Notice also that

$$c_{\Delta_k}(\ell_k) - \frac{\omega_{\ell_k}}{2} \leq c_{\Delta_k}(i) - \frac{\omega_i}{2} \quad \forall i \in V. \tag{4.2.3}$$

Indeed, this inequality follows from the definition of ℓ_k if $\ell_k \neq \alpha$. If $\ell_k = \alpha$ then $c_{\Delta_k}(\alpha) < c_{\Delta_k}(i)$ for all $i \in V$, which means that $c_{\Delta_k}(i) - c_{\Delta_k}(\alpha) \geq \delta_{\alpha i} = \frac{\omega_i}{2}$. Hence, $c_{\Delta_k}(\alpha) - \frac{\omega_\alpha}{2} = c_{\Delta_k}(\alpha) \leq c_{\Delta_k}(i) - \frac{\omega_i}{2}$. Similarly,

$$c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} \geq c_{\Delta_k}(i) + \frac{\omega_i}{2} \quad \forall i \in V. \tag{4.2.4}$$

We know from Proposition 7 that the intervals $I(i) = \{c_{\Delta_k}(i) - \frac{\omega_i}{2}, \cdots, c_{\Delta_k}(i) + \frac{\omega_i}{2} - 1\}$ define an interval coloring of G. Hence, from (4.2.3) and (4.2.4), we have

$$\begin{aligned}\chi_{int}(G) &\leq \max_{i \in V}(c_{\Delta_k}(i) + \frac{\omega_i}{2}) - \min_{i \in V}(c_{\Delta_k}(i) - \frac{\omega_i}{2}) \\ &\leq c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} - c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2}.\end{aligned}$$

So define p as the positive integer such that

$$\chi_{int}(G) = c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} - c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2} - p. \tag{4.2.5}$$

From (4.2.1) and (4.2.5), we have

$$\begin{aligned}\chi_{int}(G) - \Delta_k &\geq c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} - c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2} - p - c_{\Delta_k}(u_k) \\ &\quad + \frac{\omega_{u_k}}{2} + c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2} \\ &= \omega_{u_k} + \omega_{\ell_k} - p \end{aligned} \tag{4.2.6}$$

and from (4.2.2) and (4.2.5), we have

$$\begin{aligned}\chi_{int}(G) - \Delta_{k+1} &\leq c_{\Delta_k}(u_k) + \frac{\omega_{u_k}}{2} - c_{\Delta_k}(\ell_k) + \frac{\omega_{\ell_k}}{2} - p - c_{\Delta_k}(u_k) + c_{\Delta_k}(\ell_k) \\ &= \frac{1}{2}(\omega_{u_k} + \omega_{\ell_k}) - p \\ &\leq \frac{1}{2}(\omega_{u_k} + \omega_{\ell_k} - p). \end{aligned} \tag{4.2.7}$$

This proves the right inequality in (a) as well as (b) since (4.2.6) and (4.2.7) imply

$$\chi_{int}(G) - \Delta_k \geq 2(\chi_{int}(G) - \Delta_{k+1}).$$

Finally, notice that if $k \geq 3$, then

$$\begin{aligned}\Delta_{k-1} + \frac{\omega_{\ell_{k-2}} + \omega_{u_{k-2}}}{2} &\geq \chi_{int}(G) && \text{(by (a))} \\ &> \Delta_k && \text{(by Corollary 6)} \\ &\geq \Delta_{k-1} + \frac{\omega_{\ell_{k-1}} + \omega_{u_{k-1}}}{2} && \text{(by (a))} \end{aligned}$$

which proves (c). □

In the following, we denote W the number of different weights in G. We now give three upper bounds on the number k^* of bandwidth coloring problems which

have to be solved in the EvenReduction algorithm to determine an optimal interval coloring.

Proposition 11. *The three following inequalities define valid upper bounds on k^*:*

(i) $k^* \leq \log_2(\max_{i \in V} w_i) + 3$,

(ii) $k^* \leq 2W + 2$,

(iii) $k^* \leq |V| + 2$.

proof. To prove (i), notice that it follows from Proposition 10(a) that

$$\chi_{int}(G) - \Delta_2 \leq \frac{1}{2}(\omega_{u_2} + \omega_{\ell_2}) \leq \max_{i \in V} w_i.$$

Hence, by denoting $N = \log_2(\max_{i \in V} w_i) + 1$, we know from Proposition 10(b) that $\chi_{int}(G) - \Delta_{N+2} \leq 0$. It then follows from Corollary 6 that $k^* \leq N + 2$, which means that $k^* \leq \log_2(\max_{i \in V} w_i) + 3$.

To prove (ii), notice first that if $k^* \leq 4$ then $k^* \leq 2W + 2$ since $W \geq 1$. So

assume $k^* > 4$. Then for every iteration $k < k^* - 3$ we have

$$\begin{aligned}
max\{\omega_{\ell_k}, \omega_{u_k}\} &\geq \frac{\omega_{\ell_k} + \omega_{u_k}}{2} \\
&\geq \chi_{int}(G) - \Delta_{k+1} && \text{(by Proposition 10(a))} \\
&\geq 2(\chi_{int}(G) - \Delta_{k+2}) && \text{(by Proposition 10(b))} \\
&> 2(\Delta_{k+3} - \Delta_{k+2}) && \text{(by Corollary 6)} \\
&\geq \omega_{\ell_{k+2}} + \omega_{u_{k+2}} && \text{(by Proposition 10(a))} \\
&\geq max\{\omega_{\ell_{k+2}}, \omega_{u_{k+2}}\}
\end{aligned}$$

Since there are W different weights in G while $max\{\omega_{\ell_k}, \omega_{u_k}\} > max\{\omega_{\ell_{k+2}}, \omega_{u_{k+2}}\}$ for all $k < k^* - 3$, we know that at most $2W$ iterations are needed to reach iteration $k^* - 2$, which proves (ii).

To prove (iii), note first that if $k^* \leq 3$, then $k^* \leq |V| + 2$ as $|V| \geq 1$. So assume $k^* > 3$ and define H as the graph with vertex set $V \cup \{\gamma\}$, where γ represents both α and β, and with an edge between two vertices x and y if and only if there is an iteration k with $2 \leq k < k^* - 1$ and $\{x, y\} = \{\ell_k, u_k\}$. To each edge $\{x, y\}$ in H we associate a weight $\omega_x + \omega_y$. Observe that if $\{\ell_k, u_k\} = \{\alpha, \beta\}$, then $span(c_{\Delta_k}) - 1 = c_{\Delta_k}(u_k) - c_{\Delta_k}(\ell_k)$, which means that $k = k^*$. Hence, there is no loop at vertex γ in H. Also, we know from Proposition 10(c) that $\omega_{\ell_{k-1}} + \omega_{u_{k-1}} > \omega_{\ell_k} + \omega_{u_k}$ when $2 \leq k < k^* - 1$, which means that H contains no parallel edge.

Assume H contains a cycle C with edges $\{x_1, x_2\}, \cdots, \{x_{r-1}, x_r\}, \{x_r, x_1\}$

($r \geq 3$). Without loss of generality, we suppose that $\{x_1, x_2\}$ has maximum weight on C. We then have three iteration values i, j and k such that $\{\ell_i, u_i\} = \{x_1, x_2\}$, $\{\ell_j, u_j\} = \{x_2, x_3\}$, and $\{\ell_k, u_k\} = \{x_1, x_r\}$. Since $\{x_1, x_2\}$ has maximum weight on C, we know from Proposition 10(c) that $i < min\{j, k\}$, and we may suppose without loss of generality that $j < k$. We then have

$$
\begin{aligned}
\Delta_{k+1} &\geq \Delta_k + \frac{\omega_{x_1} + \omega_{x_r}}{2} && \text{(by Proposition 10(a))} \\
&\geq \Delta_{j+1} + \frac{\omega_{x_1} + \omega_{x_r}}{2} && \text{(by Corollary 6)} \\
&\geq \Delta_j + \frac{\omega_{x_2} + \omega_{x_3}}{2} + \frac{\omega_{x_1} + \omega_{x_r}}{2} && \text{(by Proposition 10(a))} \\
&\geq \Delta_{i+1} + \frac{\omega_{x_2} + \omega_{x_3}}{2} + \frac{\omega_{x_1} + \omega_{x_r}}{2} && \text{(by Corollary 6)} \\
&> \Delta_{i+1} + \frac{\omega_{x_1} + \omega_{x_2}}{2} \\
&\geq \chi_{int}(G) && \text{(by Proposition 10(a))}
\end{aligned}
$$

which contradicts Corollary 6. Hence H has no cycle, which means that it contains at most V edges. As a consequence, $k^* \leq |V| + 2$. □

Notice that the three bounds are sharp as illustrated in Fig. 4.4 with a graph having two vertices with weight 2. We have represented the successive graphs G_{Δ_k} with optimal bandwidth colorings c_{Δ_k}. The colors are the numbers close to the vertices while the weights are shown into boxes. Since $span(G_{\Delta_4}) - 1 = c_{\Delta_4}(\beta) - c_{\Delta_4}(\alpha)$, we have $k^* = 4 = \log_2(\max_{i \in V} w_i) + 3 = 2W + 2 = |V| + 2$.

Observe also that the EvenReduction algorithm could have reached the optimal solution in one less iteration since the bandwidth coloring c_{Δ_4} is also optimal for

G_{Δ_3}. Also, instead of initializing the EvenReduction algorithm with $\Delta_1 = 0$, one could set Δ_1 equal to any lower bound on $\chi_{int}(G)$. For example, one could determine a maximal (inclusion wise) clique (i.e., a set of pairwise adjacent vertices) in G and set Δ_1 equal to the total weight of this clique. In the above example, the unique maximal clique contains vertices v and w for a total weight of 4. By setting $\Delta_1 = 4$, the EvenReduction algorithm could skip the first three iterations, and would find the optimal interval coloring by solving only one bandwidth coloring problem.

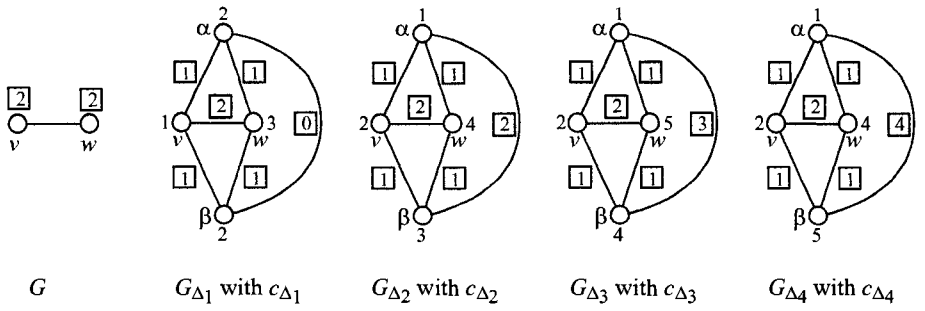


Figure 4.4: The three upper bounds on k^* are sharp.

4.3 Computational experiments

In this section, we report computational experiments in order to compare the CPU time needed to solve the interval coloring problem using the algorithm in [2] with the total CPU time needed to solve the series of bandwidth coloring problems using the algorithm in [7]. We have considered two test sets. The first experiments are performed on random graphs $G_{n,p,q}$ already used in [2] and [3]. Given a positive integer n, a real number $p \in [0, 1]$, and a positive real number q, a random graph $G_{n,p,q}$ contains n vertices, all $n(n-1)/2$ ordered pairs of vertices have a probability p of being linked by an edge, and the weight ω_i of a vertex i is generated according

to an independent truncated Poisson variable with parameter q, i.e., the probability that $\omega_i = m$ for some positive integer m is set equal to $\frac{q^m}{(e^q-1)m!}$.

The second set of instances used for the experiments are the DIMACS benchmark graphs, which come from various sources. For a detailed description of these instances, the reader can refer to `http://mat.gsia.cmu.edu/COLOR04`.

We call CS-Interval the exact algorithm proposed by Čangalović and Schreuder in [2] for finding an optimal interval coloring. It is based on the Branch-and-Bound principle. An initial lower bound on $\chi_{int}(G)$ is obtained by determining a clique of maximum total weight, using a variation of the algorithm proposed in [9]. Also, an initial upper bound on $\chi_{int}(G)$ is obtained by using the heuristic algorithm proposed in [3]. The exact algorithm proposed by Prestwich in [7] for the bandwidth coloring will be called P-Bandwidth. It combines a local search algorithm with a backtracking technique that uses constraint propagation.

Let LB denote the above mentioned lower bound on $\chi_{int}(G)$, and let G' be the graph obtained from G by multiplying every weight ω_i by 2 (see Step (1) of the GeneralReduction algorithm). Since Δ_1 can be set equal to any lower bound on $\chi_{int}(G') = 2\chi_{int}(G)$, we use $\Delta_1 = 2LB$ in order to reduce the number k^* of bandwidth coloring problems which have to be solved. Notice that if $\chi_{int}(G) = LB$, then $\chi_{int}(G') = \Delta_1$ and it follows from Corollary 6 that $k^* = 1$. Otherwise, if $\chi_{int}(G) > LB$, denote $N = \log_2(\chi_{int}(G') - \Delta_1) + 1$. Using the same arguments as in the proof of Proposition 11 we get $k^* \leq N + 1 = \log_2(\chi_{int}(G) - LB) + 1$.

For all our tests, we have considered a time limit of one hour on an AMD Opteron(tm) 285/2.6 GHz Processor. It can happen that this time is not sufficient to reach an optimal solution or to prove its optimality. When the CS-Interval algorithm has to be stopped before its end, we report the number of colors used in the best interval coloring encountered during the search, which is possibly not optimal. If the GeneralReduction algorithm has to be stopped while running the P-bandwidth

algorithm, we translate the current best bandwidth coloring into an interval coloring using the correspondance described in Proposition 7 and report its number of colors.

Since it may happen that the P-Bandwidth algorithm needs more than one hour during its first run (i.e., when applied on G'_{Δ_1}), we have also tested a version of the GeneralReduction algorithm where each run of the P-Bandwidth algorithm is stopped after at most 10 million backtracks, the overall time limit being however kept equal to one hour. This allows to run the P-Bandwidth algorithm on several graphs G'_{Δ_k} with increasing Δ_k. The resulting solution of the GeneralReduction algorithm then corresponds to an upper bound on the optimal value when at least one run of the P-Bandwidth algorithm is stopped before its end.

Table 4.1 contains the results for the graphs $G_{n,p,q}$ with $n = 20, 30, 40, 50$, $p = 0.3, 0.4, 0.5$, and $q = 1, 5, 10$. Three graphs have been generated for each such triplet (n, p, q), which gives a total of 108 instances. For each type of graph, we indicate the average value of the largest vertex weight (column labeled "max ω") and the average number of different vertex weights (column labeled "W"). We also report the average value of the lower and upper bounds (columns labeled "LB" and "UB") mentioned above. For each algorithm, we report the average number of colors in the best found interval coloring (column "χ_{int}"), the number of graphs among the three tested where optimality could be proved (column "opt"), and the average CPU time in seconds for those runs that needed less than the one hour time limit. For the GeneralReduction algorithm, we also indicate the number k^* of bandwidth coloring problems which had to be solved. The last line contains the average numbers of colors produced by each algorithm.

We can observe (see column "opt") that the CS-Interval algorithm easily finds the optimum value for graphs $G_{n,p,q}$ with a small number of vertices or with a small value of q. This is not always the case for the two versions of the GeneralReduction algorithm. For example, for $n = 20$, $p = 0.5$ and $q = 10$, no instance could be solved

to optimality with the GeneralReduction algorithm, while the CS-Interval algorithm solves the three instances with an average of 30 seconds. However, the upper bounds produced by the GeneralReduction algorithm are often optimal values. For example, for $n = 50$, $p = 0.4$, $q = 1$, the CS-Interval algorithm produces three optimal solutions with an average of 13.33 colors, while the GeneralReduction algorithm with the limit of 10 million backtracks produces colorings with the same number of colors, but without any proof of optimality.

The CS-Interval algorithm has however more difficulty to solve larger graphs within the one hour time limit. While the GeneralReduction algorithm also fails in finding optimal solutions, it produces in these cases better upper bounds. For example, for $n = 50$, $p = 0.5$ and $q = 10$, the average upper bound found by the CS-Interval algorithm is 122.33, while solutions with in average 114 colors could be produced using the GeneralReduction algorithm. For the 108 instances of this first experiment, the CS-Interval algorithm produces solutions with an average of 41.08 colors. The GeneralReduction algorithm performs a little bit better since 40.37 colors are obtained with no limit on the number of backtracks in the P-Bandwidth algorithm, and 40.02 colors otherwise.

We observe that the 10 million backtracks limit typically has a positive impact on the output of the GeneralReduction algorithm. For example, for $n = 40$, $p = 0.4$ and $q = 10$, one hour is not sufficient for the P-Bandwidth algorithm to reach an optimal solution on its first run, and the best found solution uses 76.33 colors, in average. When stopping the P-Bandwidth algorithm after at most 10 million backtracks, it is possible to run it iteratively on three different graphs (with increasing Δ_k) and this allows to obtain colorings that use only 74 colors in average. This is however not always the case as illustrated by the graphs with $n = 50$, $p = 0.5$ and $q = 10$. Indeed, one run of the P-Bandwidth algorithm during one hour produces a coloring with 114 colors in average, while 115 colors are obtained with 4 runs of the P-Bandwidth algorithm with the limit of 10 million backtracks.

Table 4.1: Results for random graphs $G_{n,p,q}$

instance							CS-Interval			GeneralReduction							
										No backtrack limit				≤10 million backtracks			
n	p	q	max ω	W	LB	UB	χ_{int}	opt	CPU	χ_{int}	opt	CPU	k^*	χ_{int}	opt	CPU	k^*
20	0.3	1	3	3	7.00	7.00	7.00	3	0	7.00	3	0	0	7.00	3	0	0
30			4	3	8.67	9.00	9.00	3	0	9.00	2	0	0.67	9.00	2	20	0.67
40			3	3	9.00	11.00	10.00	3	0	10.00	0	-	1.33	10.00	0	72	2.00
50			4	4	11.33	13.00	12.00	3	0	12.33	1	0	1.00	12.00	1	57	1.67
20	0.3	5	10	8	24.67	26.67	26.00	3	0	26.00	2	0	0.67	26.00	2	46	1.00
30			11	10	28.33	31.33	29.00	3	0	29.00	2	1	1.00	29.00	2	63	1.33
40			11	8	31.33	36.67	32.33	3	2	32.67	1	6	1.00	32.33	1	66	1.00
50			10	9	30.33	43.33	36.33	3	1283	38.33	0	-	1.00	36.33	0	279	2.33
20	0.3	10	14	9	37.33	42.00	38.33	3	6	39.33	0	-	1.00	39.33	0	133	1.00
30			16	11	50.67	59.67	55.00	3	7	56.00	0	-	1.00	55.00	0	268	1.33
40			18	12	57.33	68.00	63.00	1	6	61.00	1	61	1.00	60.00	2	239	1.33
50			18	14	60.67	84.33	78.67	0	-	73.67	0	-	1.00	73.00	0	1114	3.00
20	0.4	1	4	4	10.00	10.67	10.00	3	0	10.00	3	0	0.67	10.00	3	0	0.67
30			4	4	12.67	13.33	12.67	3	0	12.67	3	0	0.67	12.67	3	0	0.67
40			3	3	11.00	13.00	11.67	3	0	12.00	1	8	1.00	11.67	1	76	2.00
50			4	4	12.67	14.67	13.33	3	6	13.67	1	548	1.00	13.33	0	82	2.00
20	0.4	5	10	8	27.00	28.00	27.00	3	0	27.00	3	0	0.33	27.00	3	0	0.33
30			10	9	35.00	40.67	36.67	3	226	37.33	0	-	1.00	37.00	1	142	1.67
40			11	10	31.67	41.67	36.33	3	487	38.33	0	-	1.00	37.00	0	254	2.33
50			9	9	38.67	54.67	47.67	0	-	49.33	0	-	1.00	48.67	0	578	3.33
20	0.4	10	15	8	51.67	55.00	53.33	3	0	54.33	1	0	0.67	53.67	1	90	0.67
30			17	11	61.67	75.33	64.67	3	93	65.00	1	40	1.00	65.00	1	247	1.33
40			15	12	59.67	83.67	78.33	0	-	76.33	0	-	1.00	74.00	0	854	3.00
50			18	14	74.33	103.67	102.00	0	-	91.33	0	-	1.00	90.67	0	982	2.67
20	0.5	1	3	3	9.00	9.67	9.67	3	0	9.67	1	0	1.33	9.67	1	32	1.33
30			3	3	12.33	13.00	12.33	3	0	12.33	3	0	0.67	12.33	3	0	0.67
40			3	3	13.00	16.00	14.33	3	0	15.00	0	-	1.00	14.33	0	122	2.33
50			4	4	15.33	20.33	17.67	3	355	18.33	0	-	1.00	18.00	0	196	3.33
20	0.5	5	10	8	34.33	36.67	34.67	2	0	35.33	1	0	0.67	35.00	1	47	0.67
30			10	9	38.33	45.33	40.67	3	85	41.33	1	8	1.00	41.33	0	188	2.00
40			9	9	37.33	50.67	45.33	1	1561	46.00	0	-	1.00	45.33	0	385	3.33
50			9	9	42.00	59.33	55.33	0	-	52.33	0	-	1.00	52.00	0	463	3.33
20	0.5	10	16	8	52.00	58.67	54.00	3	30	54.33	0	-	1.00	54.33	0	183	1.33
30			19	11	68.33	85.33	77.67	1	1419	78.00	0	-	1.00	77.00	0	762	2.33
40			16	12	73.33	111.33	104.67	0	-	95.00	0	-	1.00	96.67	0	1255	4.00
50			19	13	88.00	129.33	122.33	0	-	114.00	0	-	1.00	115.00	0	1669	4.00
average					35.17	44.50	41.08			40.37				40.02			

Table 4.2 reports computational experiments on the DIMACS benchmark graphs. These instances have up to 191 vertices and are therefore more challenging. The three first columns contain the name of the instances, their number n of vertices, and their number m of edges. The next columns are similar to those of Table 4.1 except that column "opt" is replaced with the following rule : when an algorithm does not produce a solution with a proof of optimality, we report the obtained upper bound under column "χ_{int}" with italic characters, while normal characters are used when optimality is proved. CPU times are only given for instances where the GeneralReduction algorithm stopped before the 1 hour time limit. We do not report any result for instances R50_1g, R50_1gb and GEOMx with x = 20, 20a, 20b, 30, 30a, 30b, 40, 40a, 40b, 50, 50a, 60, 70, 70a, 80, 90b, 100a, 110a, 110b, since the lower bound LB and the upper bound UB are equal on these graphs (hence there is no need to use a Branch-and-Bound procedure).

The CS-Interval algorithm is able to solve 9 instances to optimality within the one hour time limit, while a proof of optimality is obtained 13 times for each version of the GeneralReduction algorithm. The GEOM instances seem to be easier for the GeneralReduction algorithm since 11 of the 13 instances are solved to optimality, while the CS-Interval algorithm solves only two of them. A deeper analysis indicates that the 6 instances which are solved to optimality by the CS-Interval algorithm and not by the GeneralReduction algorithm all have $\chi_{int}(G) > LB$, while the 10 instances which are solved to optimality by the GeneralReduction algorithm and not the CS-Interval algorithm all have $\chi_{int}(G) = LB$. Hence the GeneralReduction algorithm seems to be particularly effective in proving optimality when the lower bound equals the optimal value. This is confirmed by the results in Table 4.1 since there are only 5 triplets (n, p, q) for which we know that $\chi_{int}(G_{n,p,q}) = LB$ (namely $(20, 0.3, 1), (20, 0.4, 1), (30, 0.4, 1), (20, 0.4, 5)$ and $(30, 0.5, 1)$), and the GeneralReduction algorithm is able to obtain a proof of optimality (i.e., opt=3) only on these instances.

Table 4.2: Results for DIMACS benchmark graphs

instance							CS-Interval		GeneralReduction					
									No backtrack limit			≤10 million backtracks		
name	n	m	max ω	W	LB	UB	χ_{int}	CPU	χ_{int}	CPU	k^*	χ_{int}	CPU	k^*
DSJC125.1gb	125	736	20	20	67	86	*82*	-	*73*	-	1	*74*	928	2
DSJC125.1g	125	736	5	5	19	23	19	7	19	83	1	19	83	1
DSJC125.5gb	125	3891	20	20	125	246	*246*	-	*239*	-	1	*237*	-	4
DSJC125.5g	125	3891	5	5	40	72	*72*	-	*68*	-	1	*69*	584	3
DSJC125.9gb	125	6961	20	20	425	608	*608*	-	*608*	-	1	*608*	-	1
DSJC125.9g	125	6961	5	5	122	166	*166*	-	*163*	-	1	*164*	-	4
GEOM50b	50	249	3	3	17	18	17	0	17	0	1	17	0	1
GEOM60a	60	339	10	10	65	66	*66*	-	65	56	1	65	56	1
GEOM60b	60	366	3	3	22	23	22	508	22	2	1	22	2	1
GEOM70b	70	488	3	3	22	23	*23*	-	22	12	1	22	12	1
GEOM80a	80	612	10	10	68	72	*72*	-	*69*	-	1	*69*	1065	2
GEOM80b	80	663	3	3	25	26	*26*	-	25	1	1	25	1	1
GEOM90a	90	789	10	10	65	70	*69*	-	65	0	1	65	0	1
GEOM90	90	441	10	10	51	52	*52*	-	51	9	1	51	9	1
GEOM100b	100	1050	3	3	30	31	*31*	-	30	0	1	30	0	1
GEOM100	100	547	10	10	60	61	*61*	-	60	0	1	60	0	1
GEOM110	110	638	10	10	62	67	*67*	-	62	46	1	62	46	1
GEOM120a	120	1434	10	10	93	98	*98*	-	*98*	-	1	*98*	2661	3
GEOM120b	120	1491	3	3	34	35	*35*	-	34	10	1	34	10	1
GEOM120	120	773	10	10	63	68	*68*	-	*67*	-	1	*65*	1645	2
myciel5gb	47	236	20	20	37	65	53	0	*57*	-	1	*53*	838	3
myciel5g	47	236	5	5	10	19	17	1	*19*	-	1	*17*	134	3
myciel6gb	95	755	20	20	39	92	*82*	-	*84*	-	1	*79*	2243	4
myciel6g	95	755	5	5	10	25	20	308	*21*	-	1	*20*	272	3
myciel7gb	191	2360	20	20	40	98	*93*	-	*93*	-	1	*90*	2185	3
myciel7g	191	2360	5	5	10	28	*28*	-	*24*	-	1	*24*	541	4
queen10_10gb	100	1470	20	20	136	159	*159*	-	*147*	-	1	*146*	-	2
queen10_10g	100	1470	5	5	38	43	*40*	-	*39*	-	1	*39*	618	2
queen11_11gb	121	1980	20	19	140	170	*170*	-	*168*	-	1	*165*	2335	3
queen11_11g	121	1980	5	5	41	48	*44*	-	*44*	-	1	*44*	618	2
queen12_12gb	144	2596	20	20	163	192	*192*	-	*179*	-	1	*180*	-	3
queen12_12g	144	2596	5	5	42	52	*49*	-	*49*	-	1	*49*	536	3
queen8_8gb	64	728	20	20	113	120	*117*	-	113	220	1	113	220	1
queen8_8g	64	728	5	5	28	34	*29*	-	*32*	-	1	*30*	276	3
queen9_9gb	81	1056	20	20	135	157	*154*	-	*148*	-	1	*145*	-	2
queen9_9g	81	1056	5	5	35	39	35	615	*36*	-	1	*36*	236	1
R100_1gb	100	509	20	20	56	78	*78*	-	*70*	-	1	*65*	800	2
R100_1g	100	509	5	5	15	19	17	3	*19*	-	1	*18*	132	2
R100_5gb	100	2456	20	20	132	222	*222*	-	*208*	-	1	*211*	-	4
R100_5g	100	2456	5	5	35	58	*58*	-	*54*	-	1	*55*	461	3
R100_9gb	100	4438	20	20	390	512	*512*	-	*512*	-	1	*512*	-	1
R100_9g	100	4438	5	5	108	140	*140*	-	*134*	-	1	*134*	1422	4
R50_5gb	50	612	20	19	98	131	*130*	-	*118*	-	1	*119*	1993	4
R50_5g	50	612	5	5	27	34	*32*	-	*34*	-	1	*33*	338	4
R50_9gb	50	1092	19	18	228	264	*264*	-	*264*	-	1	*264*	-	1
R50_9g	50	1092	5	5	64	73	*73*	-	*69*	-	1	*68*	637	4
R75_1gb	70	251	20	20	53	69	*63*	-	*57*	-	1	*57*	303	1
R75_1g	70	251	5	5	14	19	16	0	*17*	-	1	*16*	151	2
R75_5gb	75	1407	20	20	114	184	*184*	-	*172*	-	1	*171*	2996	4
R75_5g	75	1407	5	5	31	50	*50*	-	*46*	-	1	*46*	386	3
R75_9gb	75	2513	20	20	298	393	*393*	-	*393*	-	1	*393*	-	1
R75_9g	75	2513	5	5	85	104	*104*	-	*102*	-	1	*101*	1150	4
average					81.53	107.73	106.12		103.44			102.87		

We next observe from Table 4.2 that the CS-Interval algorithm produces solutions with an average of 106.12 colors while, again, the GeneralReduction algorithm performs better in average since 103.44 colors are found with no limit on the number of backtracks in the P-Bandwidth algorithm, and 102.87 colors otherwise. We also notice that if we except four instances (namely queen8_8g, queen9_9g, R100_1g and R50_5g), the number of colors produced by the GeneralReduction algorithm with the limit of 10 million backtracks is never larger than the number of colors obtained with the CS-Interval algorithm, and it is even strictly smaller for 34 of the 52 instances.

Notice also that the GeneralReduction algorithm with the 10 million backtracks limits has required less than one hour CPU for 42 instances. This means that the P-Bandwidth algorithm has produced a coloring c_{Δ_k} with $|c_{\Delta_k}(\beta) - c_{\Delta_k}(\alpha)| = \Delta_{k+1}$ within the one hour time limit. As mentioned above, this does not mean that the resulting solution is optimal since previous runs of the P-Bandwidth algorithm with smaller values of Δ_k have possibly reached the 10 million backtracks limit.

4.4 Conclusion

We have shown that an optimal solution of the interval coloring problem can be obtained by solving a series of bandwidth coloring problems. Since both the interval and the bandwidth coloring problems are NP-hard, such a reduction could appear as useless. However, we have shown that when a lower bound LB on $\chi_{int}(G)$ can be obtained and it has to be proved that $\chi_{int}(G) = LB$, then the proposed reduction is particularly efficient since it allows to solve larger instances than those solved by the exact CS-Interval algorithm. Notice that $k^* = 1$ in these cases, which means that the solution of the interval coloring problem is reduced to the solution of exactly one bandwidth coloring problem.

We have shown that for large instances, where the optimal solution can hardly be obtained in a reasonable amount of time, the proposed reduction helps producing

better upper bounds on $\chi_{int}(G)$. An average improvement of more than 3% could be observed on the DIMACS benchmark graphs. Also, we have shown that by fixing a limit on the number of backtracks when solving each bandwidth coloring problem with a Branch-and-Bound algorithm, it is possible to use increasing values of Δ_k and to typically obtain better upper bounds on $\chi_{int}(G)$.

BIBLIOGRAPHY

[1] Čangalović, M. (1989). Some new combinatorial optimization algorithms applied to timetabling problems. Ph.D. thesis, University of Belgrade, Serbia (in Serbian).

[2] Čangalović, M., Schreuder, J.A.M. (1991). Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths. *European Journal of Operational Research*, 51, 248–258.

[3] Clementson, A.T., Elphick, C.H. (1983). Approximate colouring algorithms for composite graphs. *Journal of Operational Research Society*, 34, 503–509.

[4] Deuber, W.A., Zhu, X. (1996). Circular colorings of weighted graphs. *Journal of Graph Theory*, 24, 365–376.

[5] Fabri, F. (1979). Automatic storage optimization. *ACM SIGPLAN Notices: Proceedings of the 1979 SIGPLAN symposium on Compiler construction*, 14/8:83–91.

[6] Garey, M.R., Johnson, D.S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W.H. Freman and Company, NY.

[7] Prestwich, S. (2002). Constrained bandwidth multicoloration neighborhoods. *Proceedings of Computational Symposium on Graph Coloring and its Generalizations*,Ithaca, NY, USA, pp. 126–133.

[8] Punter, A. (1976). *Systems for timetabling by computer based on graph coloring.* Ph.D. Thesis, C.N.A.A., Hatfield Polytechnic.

[9] Sakaki, T., Nakashima, K., Hattori, Y. (1976). Algorithms for finding in the lump both bounds of the chromatic number of a graph. *The Computer Journal*, 19, 329–332.

CHAPITRE 5 : ABOUT EQUIVALENT INTERVAL COLORINGS OF WEIGHTED GRAPHS

Soumis dans *Discrete Applied Mathematics, Graph Optimization VI - Special Issue* le 21 décembre 2007.

About Equivalent Interval Colorings of Weighted Graphs

Mathieu Bouchard [1] **Mirjana Čangalović** [2] **Alain Hertz** [1]

[1] *École Polytechnique and GERAD, Montréal, Canada*

{Mathieu.Bouchard, Alain.Hertz}@gerad.ca

[2] *Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia*

canga@fon.bg.ac.yu

Abstract

Given a graph $G = (V, E)$ with strictly positive integer weights ω_i on the vertices $i \in V$, a k-interval coloring of G is a function I that assigns an interval $I(i) \subseteq \{1, \cdots, k\}$ of ω_i consecutive integers (called colors) to each vertex $i \in V$. If two adjacent vertices x and y have common colors, i.e. $I(i) \cap I(j) \neq \emptyset$ for an edge $[i, j]$ in G, then the edge $[i, j]$ is said *conflicting*. A k-interval coloring without conflicting edges is said *legal*. The interval coloring problem (ICP) is to determine the smallest integer k, called *interval chromatic number* of G and denoted $\chi_{int}(G)$, such that there exists a legal k-interval coloring of G. For a fixed integer k, the k-interval graph coloring problem (k-ICP) is to determine a k-interval coloring of G with a minimum number of conflicting edges. The ICP and k-ICP generalize *classical vertex coloring problems* where a single color has to be assigned to each vertex (i.e., $\omega_i = 1$ for all vertices $i \in V$).

Two k-interval colorings I_1 and I_2 are said *equivalent* if there is a permutation π of the integers $1, \cdots, k$ such that $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$ for all vertices $i \in V$. As for classical vertex coloring, the efficiency of algorithms that solve the ICP or the k-ICP can be increased by avoiding considering equivalent k-interval colorings. To this purpose, we define and prove a necessary and sufficient condition for the equivalence of two k-interval colorings. We then show how a simple tabu search algorithm for the k-ICP can possibly be improved by forbidding the visit of equivalent solutions.

5.1 Introduction

Given a graph $G = (V, E)$ with vertex set V and edge set E, the classical graph coloring problem is to assign a color to each vertex so that no two adjacent vertices have the same color and the total number of different colors is minimized. This is one of the most studied NP-hard combinatorial optimization problems [8] with various practical applications [15]. A number of different variations and generalizations of the classical graph coloring problem arise when modeling and solving real-life problems. For example, the number of colors assigned to a vertex can be more than one, and conditions can be imposed on the colors assigned to the vertices.

One such generalization is the so-called interval coloring problem of a vertex-weighted graph [11] where a strictly positive integer weight ω_i is associated with each vertex $i \in V$, and an interval of ω_i consecutive integers must be assigned to each vertex $i \in V$ such that the intervals assigned to adjacent vertices are disjoint. More formally, let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, and with strictly positive integer weights ω_i on the vertices $i \in V$. A k-interval coloring of G is a function I that assigns an interval $I(i) \subseteq \{1, \cdots, k\}$ of ω_i consecutive integers (called colors) to each vertex $i \in V$. Without loss of generality, we will always assume that a k-interval coloring of G uses all colors in $\{1, \cdots, k\}$. If two adjacent vertices x and y have common colors, i.e. $I(i) \cap I(j) \neq \emptyset$ for an edge $[i, j]$ in G, then the edge $[i, j]$ is said *conflicting*. A k-interval coloring without conflicting edges is said *legal*. The interval coloring problem (ICP) is to determine the smallest integer k, called *interval chromatic number* of G and denoted $\chi_{int}(G)$, such that there exists a legal k-interval coloring of G. The special case with $\omega_i = 1$ for all vertices $i \in V$ is equivalent to the classical graph coloring problem, and a k-interval coloring is simply called k-coloring in this case. For illustration, a legal 6-interval coloring of a graph G is represented in Figure 5.1, where the numbers into boxes correspond to weights on vertices. Note that $\chi_{int}(G) = 6$ for this graph since the total weight of the edge $[a, b]$ or of the triangle with vertices b, d, e is equal to 6.

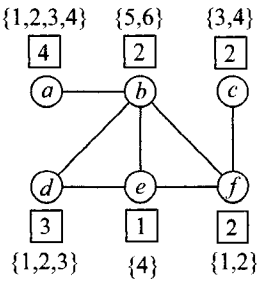


Figure 5.1: A legal 6-interval coloring I of a graph G.

For a fixed integer k, the k-interval graph coloring problem (k-ICP) is to determine a k-interval coloring of G with a minimum number of conflicting edges. If the minimum value is zero, this means that G admits a legal k-interval coloring, hence $\chi_{int}(G) \leq k$.

The ICP has a fairly long history dating back, at least to the 1970s. For example, Stockmeyer showed in 1976 that the interval-coloring problem is NP-hard, even when restricted to interval graphs and vertex weights in $\{1, 2\}$ (see problem SR2 in [8]). In 1976, Punter has formulated and solved a school timetabling problem with non-preemptive multiple period lessons using an interval coloring model. Another early application of the interval coloring problem was in the compile-time memory-allocation problem [5].

While the ICP is NP-hard, it can be solved in polynomial time for special classes of graphs. For example, if G is a clique then, $\chi_{int}(G)$ is equal to $\sum_{i \in V} \omega_i$, while for a bipartite graph G, we have $\chi_{int}(G) = \max_{[i,j] \in E}\{\omega_i + \omega_j\}$. More general graphs G for which $\chi_{int}(G)$ can be computed in polynomial time are studied in [11, 20].

Upper bounds on the interval chromatic number $\chi_{int}(G)$ are studied in [20], while general upper and lower bounds on $\chi_{int}(G)$ are given in [14] when vertices possibly have forbidden colors. An exact algorithm for the ICP is proposed in [4] and used to solve a real-life timetabling problem with multiple period lessons. Approximation

algorithms are known for special classes of graphs such as interval graphs [2, 9] or chordal graphs [16]. Heuristic algorithms for the ICP are proposed for example in [3, 1].

In the classical graph coloring problem, every k-coloring is equivalent, up to a permutation of the colors, to $k! - 1$ other k-colorings. In order to increase the efficiency of graph coloring algorithms, it is important to avoid visiting equivalent k-colorings when exploring the search space. A solution to the classical k-coloring problem is in fact a partition of the vertex set into k subsets called *color classes*, and the total number of non equivalent k-colorings is equal to the number of possible partitions of the vertex set into k subsets. Such considerations have inspired many researchers, including Galinier and Hao [6] who have designed a very effective genetic algorithm for the classical graph coloring problem in which new k-colorings are created from a population of k-colorings by combining color classes of two parents instead of copying color assignments. Also, the most effective local search algorithms for classical graph coloring generate neighbor k-colorings by moving a vertex from a color class to another [7].

In the next section we generalize the above equivalence relation to k-interval colorings. We then prove a necessary and sufficient condition for the equivalence of two k-interval colorings. Such a condition makes it easy to recognize equivalent k-interval colorings. We will use the following terminology. A *clique* in a graph $G = (V, E)$ is a subset $W \subseteq V$ of pairwise adjacent vertices. An *interval graph* [12, 11] is the intersection graph of a set of intervals. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intervals that intersect. Subsets $C_1, \cdots, C_p$ of V define a *cover* of V if $\bigcup_{i=1}^{p} C_i = V$. Moreover, if the sets C_i of the cover are mutually disjoint, they define a *partition* of V.

5.2 Equivalent k-interval colorings

Intuitively, two k-interval colorings are equivalent if one can be obtained from the other by permuting the colors $1, \cdots, k$. More formally, the equivalence of k-interval colorings can be defined as follows.

Definition 1. *Two k-interval colorings I_1 and I_2 are said equivalent if there is a permutation π of the integers $1, \cdots, k$ such that $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$ for all vertices $i \in V$.*

Note that given a k-interval coloring I of graph $G = (V, E)$ and a permutation π of the integers $1, \cdots, k$, it may happen that $\bigcup_{\ell \in I(i)} \pi(\ell)$ is not an interval for some vertex $i \in V$. For example, considering the graph of Figure 5.1, permutation π with $\pi(1) = 6$, $\pi(6) = 1$ and $\pi(\ell) = \ell$ for $\ell \neq 1, 6$ gives colors $2, 3, 4$ and 6 to vertex a, and this is not an interval.

For the classical graph coloring problem (i.e., when $\omega_i = 1$ for all $i \in V$), Definition 1 means that two k-colorings are equivalent if their corresponding partition into color classes are identical. A concept similar to color classes in the case of interval coloring is what we call *interval color classes*, with the following formal definition.

Definition 2. *Given a k-interval coloring I of a graph $G = (V, E)$, a subset $W \subseteq V$ of vertices is an interval color class for I if*

(a) $\bigcap_{i \in W} I(i) \neq \emptyset$, *and*

(b) $\bigcap_{i \in W} I(i) \cap I(j) = \emptyset$ *for all* $j \notin W$.

The above definition can be interpreted in terms of graphs. Indeed, given an k-interval coloring I of a graph $G = (V, E)$, let $H_{G,I}$ be the interval graph with vertex set V and where two vertices i and j are linked by an edge if and only if $I(i) \cap I(j) \neq \emptyset$. Then, the interval color classes for I correspond to the maximal

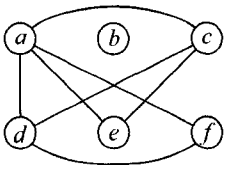


Figure 5.2: The interval graph $H_{G,I}$ associated with the k-interval coloring of Figure 5.1.

cliques in $H_{G,I}$. For example, the graph of Figure 5.2 is the interval graph $H_{G,I}$ associated with the k-interval coloring of Figure 5.1. It contains 4 maximal cliques (i.e. interval color classes), namely $W_1 = \{a, c, d\}$, $W_2 = \{a, c, e\}$, $W_3 = \{a, d, f\}$ and $W_4 = \{b\}$.

When $\omega_i = 1$ for all $i \in V$ (i.e., for the classical graph coloring problem), the interval graph $H_{G,I}$ is made of vertex-disjoint cliques, each one corresponding to a color class. Observe that the color classes in classical graph coloring induce a partition of the vertex set, while the interval color classes for a k-interval coloring induce a cover of the vertex set, which means that some vertices possibly belong to several interval color classes. In the example of Figure 5.2, vertex a belongs to three different interval color classes, and vertex c belongs to two of them.

Since two k-colorings in classical graph coloring are equivalent if and only if they induce the same partition of the vertex set into color classes, it is tempting to think that two k-interval colorings I_1 and I_2 of a graph G are equivalent if and only if they have exactly the same interval color classes, i.e. if their associated interval graphs H_{G,I_1} and H_{G,I_2} are equal. Figure 5.3 illustrates with an example that such a statement is not correct. Indeed, the two 7-interval colorings I_1 and I_2 on this figure have the same associated interval graph. However, if I_1 and I_2 were equivalent, then $I_2(f) = \{1, 2\} = \{\pi(3), \pi(4)\}$, which means that $\{1, 2\} \subset I_2(d)$, a contradiction.

Hence, in order to determine whether two k-interval colorings of a graph $G = (V, E)$ are equivalent, it is not sufficient to compare their corresponding cover of V

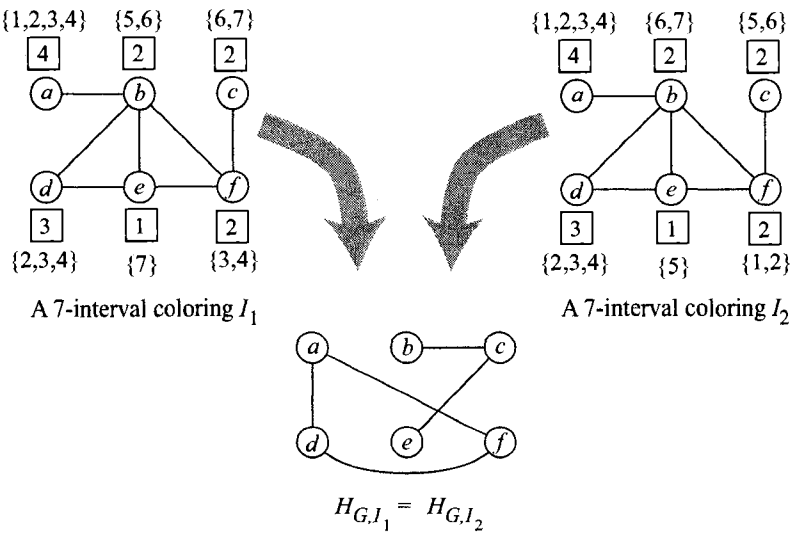


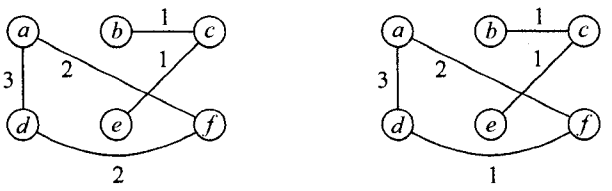
Figure 5.3: Two non-equivalent 7-interval colorings of a graph G, and their identical associated interval graph.

with interval color classes. For a k-interval coloring I of G, let us associate a weight $|I(i) \cap I(j)|$ to each edge $[i, j]$ in the interval graph $H_{G,I}$. The following Theorem states that two k-interval colorings I_1 and I_2 of a graph G are equivalent if and only if the corresponding weighted graphs H_{G,I_1} and H_{G,I_2} are identical (i.e., they have the same edge set and the same weights on the edges). For example, the two weighted interval graphs associated with the 7-interval colorings of Figure 5.3 are represented in Figure 5.4. Since the weight of the edge $[d, f]$ is 2 in H_{G,I_1} and 1 in H_{G,I_2}, the two 7-interval colorings are not equivalent.

Theorem 5.2.1. *Two k-interval colorings I_1 and I_2 of a graph $G = (V, E)$ are equivalent if and only if*

$$|I_1(i) \cap I_1(j)| = |I_2(i) \cap I_2(j)| \text{ for all } i, j \text{ in } V \tag{5.1}$$

We first prove the following Lemma about k-interval colorings satisfying Property



H_{G,I_1} H_{G,I_2}

Figure 5.4 – The two weighted interval graphs associated with the 7-interval colorings of Figure 5.3.

(5.1) of Theorem 5.2.1. For a subset $W \subseteq V$ of vertices and a k-interval coloring I, we denote $I(W) = \bigcap_{i \in W} I(i)$.

Lemma 1. *Let I_1 and I_2 be two k-interval colorings of a graph $G = (V, E)$ satisfying Property (5.1), and let $W \subseteq V$ be a subset of vertices. Then*
(a) $|I_1(W)| = |I_2(W)|$, and
(b) W is an interval color class for I_1 if and only if it is an interval color class for I_2.

Démonstration. We first prove (a). If W contains a single vertex i (i.e. $W = \{i\}$), then $\omega_i = |I_1(W)| = |I_2(W)|$. So assume $|W| \geq 2$.

- If $|I_1(W)| = 0$ then there exist at least two vertices i and j in W such that $I_1(i) \cap I_1(j) = \emptyset$. By Property (5.1), we then have $|I_2(i) \cap I_2(j)| = 0$, which means that $|I_2(W)| = 0$.
- If $|I_1(W)| > 0$, then $I_1(W)$ is an integer interval (since $I_1(i)$ is an integer interval for all $i \in V$), and there are at least two vertices u and v in W such that $I_1(u) \cap I_1(v) = I_1(W)$. By Property (5.1), we have $|I_2(u) \cap I_2(v)| = |I_1(W)|$, which means that $|I_2(W)| \leq |I_1(W)|$ since $I_2(W) \subseteq I_2(u) \cap I_2(v)$.

In all cases, we have $|I_2(W)| \leq |I_1(W)|$. By permuting the roles of I_1 and I_2, the same proof gives $|I_1(W)| \leq |I_2(W)|$. Hence, $|I_2(W)| = |I_1(W)|$.

To prove (b), assume that W is an interval color class for one of the two k-interval colorings, say I_1. Then, by definition, we have $I_1(W) \neq \emptyset$, and it follows from (a) that $I_2(W) \neq \emptyset$. If there exists a vertex $u \notin W$ such that $I_2(W \cup \{u\}) \neq \emptyset$, then we know from (a) that $I_1(W \cup \{u\}) \neq \emptyset$, which means that W is not an interval color class for I_1, a contradiction. In summary, $I_2(W) \neq \emptyset$ and $I_2(W \cup \{u\}) = \emptyset$ for all $u \notin W$, which means that W is also an interval color class for I_2. □

We now prove that Property (5.1) is a necessary and sufficient condition for the equivalence of two k-interval colorings.

Proof of Theorem 5.2.1

Let I_1 and I_2 be two equivalent k-interval colorings of a graph $G = (V, E)$. By definition 1, there exists a permutation π of the integers $1, \cdots, k$ such that $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$ for all vertices $i \in V$. Hence, for every i and j in V and for every $\ell \in \{1, \cdots, k\}$ we have $\ell \in I_1(i) \cap I_1(j)$ if and only if $\pi(\ell) \in I_2(i) \cap I_2(j)$, which means that $|I_1(i) \cap I_1(j)| = |I_2(i) \cap I_2(j)|$ for all i and j in V. Property (5.1) is therefore a necessary condition for the equivalence of two k-interval colorings.

We now prove that Property (5.1) is also a sufficient condition. The proof is by induction on the number k of colors used in I_1 and I_2. For $k = 1$, we have $I_1(i) = I_2(i) = \{1\}$ for all vertices $i \in V$ (since a k-interval coloring uses all colors in $\{1, \cdots, k\}$). Hence, permutation π with $\pi(1) = 1$ defines the equivalence between I_1 and I_2.

So, assume that $k > 1$ and Property (5.1) is a sufficient condition for the equivalence of two ℓ-interval colorings for all $\ell = 1, \cdots, k-1$. Consider any interval color class W for I_1. We know from Lemma 1 that $|I_1(W)| = |I_2(W)|$ and W is also an interval color class for I_2. So let π_1 be a bijective mapping from $I_1(W)$ to $I_2(W)$ (i.e., $\bigcup_{\ell \in I_1(W)} \pi_1(\ell) = I_2(W)$). For all vertices $i \in V$ and $r = 1, 2$ define

$$I'_r(i) = \begin{cases} I_r(i) - I_r(W) & \text{if } i \in W \\ I_r(i) & \text{if } i \in V - W \end{cases}$$

and

$$\omega'_i = \begin{cases} \omega_i - |I_1(W)| & \text{if } i \in W \\ \omega_i & \text{if } i \in V - W \end{cases}$$

Since W is an interval color class for I_r $(r = 1, 2)$, we have $\bigcup_{i \in V} I'_r(i) = \{1, \cdots, k\} - I_r(W)$ and $|I'_r(i)| = \omega'_i$ for all $i \in V$ and $r = 1, 2$. Note that $I'_r(i)$ is not necessarily an interval. Indeed, if the smallest color in $I_r(W)$ is strictly larger than the smallest color in $I_r(i)$ while the largest color in $I_r(W)$ is strictly smaller than the largest color in $I_r(i)$, then $I'_r(i)$ is the union of two integer intervals. So, let f_r be a function that relabels the colors in $\{1, \cdots, k\} - I_r(W)$ from 1 to $k - |I_r(W)|$ so that $f_r(i) < f_r(j)$ if and only if $i < j$ and define $I''_r(i) = \bigcup_{\ell \in I'_r(i)} f_r(\ell)$. The sets $I''_r(i)$ are integer intervals for all $i \in V$ and $r = 1, 2$. Note that if $I_r(i) = I_r(W)$ for a vertex $i \in W$, then $\omega'_i = 0$ and $I''_r(i)$ is empty.

Let $G' = (V', E')$ be the weighted graph obtained from G by removing all vertices with $\omega'_i = 0$, and by assigning weight ω'_i to all vertices $i \in V'$. By denoting $k' = k - |I_1(W)| = k - |I_2(W)|$, we have shown that I''_1 and I''_2 are two k'-interval colorings of G' with $k' < k$. In order to use the induction hypothesis, we now show that I''_1 and I''_2 satisfy Property (5.1).

- If i and j are two vertices in $V' \cap W$, then $I_r(W) \subseteq I_r(i) \cap I_r(j)$ for $r = 1, 2$, which means that $|I''_r(i) \cap I''_r(j)| = |I_r(i) \cap I_r(j)| - |I_r(W)|$. Since $|I_1(i) \cap I_1(j)| = |I_2(i) \cap I_2(j)|$ and $|I_1(W)| = |I_2(W)|$, we have $|I''_1(i) \cap I''_1(j)| = |I''_2(i) \cap I''_2(j)|$.
- If i and j are two vertices in V' with at least one not in W, then $I_r(W) \cap I_r(i) \cap I_r(j) = \emptyset$ for $r = 1, 2$, which means that $|I''_r(i) \cap I''_r(j)| = |I_r(i) \cap I_r(j)|$. Since $|I_1(i) \cap I_1(j)| = |I_2(i) \cap I_2(j)|$, we have $|I''_1(i) \cap I''_1(j)| = |I''_2(i) \cap I''_2(j)|$.

By induction hypothesis, we know that there exists a permutation π_2 of the colors in $\{1, \cdots, k'\}$ such that $\ell \in I''_1(i)$ if and only if $\pi_2(\ell) \in I''_2(i)$ for all vertices $i \in V'$.

Consider finally permutation π of the colors in $1, \cdots, k$ such that

$$\pi(\ell) = \begin{cases} \pi_1(\ell) & \text{if } \ell \in I_1(W) \\ f_2^{-1}(\pi_2(f_1(\ell))) & \text{if } \ell \in \{1, \cdots, k\} - I_1(W) \end{cases}$$

For a vertex $i \in V$ and a color $\ell \in I_1(i)$, we have proved that

- if $\ell \in I_1(W)$, then $\pi(\ell) = \pi_1(\ell) \in I_2(W) \subseteq I_2(i)$
- if $\ell \notin I_1(W)$, then $f_1(\ell) \in I_1''(i)$. Since $\pi_2(f_1(\ell)) \in I_2''(i)$, we have $\pi(\ell) = f_2^{-1}(\pi_2(f_1(\ell))) \in I_2(i)$.

In summary, we have $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$, which proves that I_1 and I_2 are equivalent. □

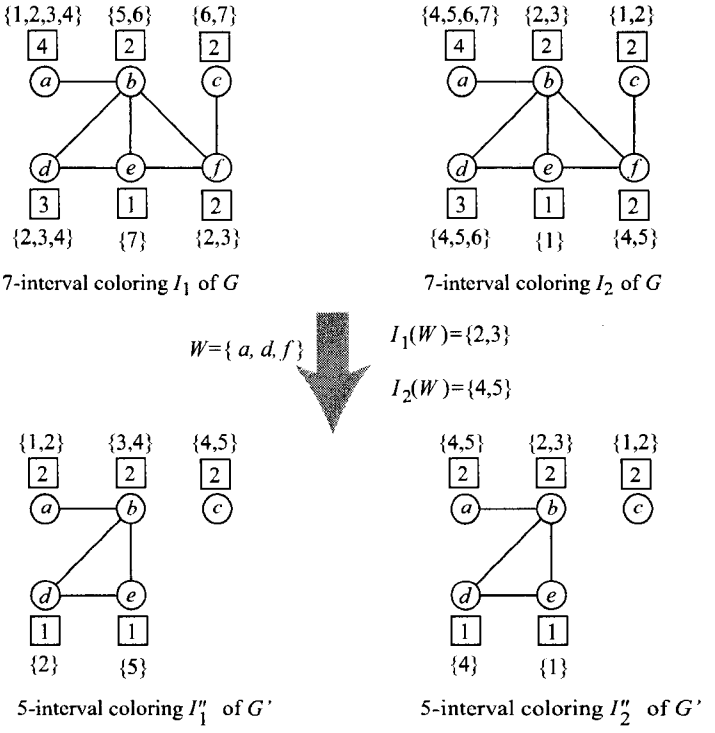


Figure 5.5 – Illustration of the proof of Theorem 5.2.1.

An illustration of the above construction of permutation π for two equivalent k-interval colorings is given in Figure 5.5. Vertex set $W = \{a, d, f\}$ is an interval

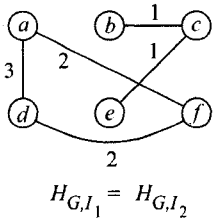


Figure 5.6 – The weighted interval graph associated with the 7-interval colorings of Figure 5.5.

color class for both 7-interval colorings I_1 and I_2 of G. We have $I_1(W) = \{2, 3\}$ and $I_2(W) = \{4, 5\}$. We can therefore consider π_1 such that $\pi_1(2) = 4$ and $\pi_1(3) = 5$. Hence, $f_1(1) = 1, f_1(4) = 2, f_1(5) = 3, f_1(6) = 4, f_1(7) = 5$ and $f_2(1) = 1, f_2(2) = 2, f_2(3) = 3, f_2(6) = 4, f_2(7) = 5$. Since $\omega'_f = 0$, vertex f does not belong to G'. All vertices in G' have the same weight as in G, except a and d for which there is reduction of two units. While $I'_1(a) = \{1, 4\}$ is not an interval, $I''_1(a) = \{f_1(1), f_1(4)\} = \{1, 2\}$. The two 5-interval colorings I''_1 and I''_2 of G' are equivalent, which can be observed with permutation π_2 such that $\pi_2(1) = 5, \pi_2(2) = 4, \pi_2(3) = 3, \pi_2(4) = 2$ and $\pi_2(5) = 1$. A proof of the equivalence of I_1 and I_2 in G is provided by permutation π with $\pi(1) = 7, \pi(2) = 4, \pi(3) = 5, \pi(4) = 6, \pi(5) = 3, \pi(6) = 2$ and $\pi(7) = 1$. For example, $\pi(4) = f_2^{-1}(\pi_2(f_1(4))) = f_2^{-1}(\pi_2(2)) = f_2^{-1}(4) = 6$. The weighted interval graph associated with the two 7-interval colorings of Figure 5.5 is represented in Figure 5.6.

5.3 Comparison of two algorithms for the ICP

In order to illustrate how the theoretical results of the previous section can help in the design of efficient algorithms for the ICP, we have developed two tabu search algorithms, the second one being based on Theorem 5.2.1 for avoiding the visit of equivalent solutions. We will show on a limited set of instances that the second tabu

search algorithm possibly finds better solutions than the first one. The computational experiments are not meant to be exhaustive, but rather indicative and should help to orient future research on the development of more elaborate algorithms for the ICP.

5.3.1 Two tabu search algorithms for the ICP

Let S be the set of solutions to a combinatorial optimization problem, and f a function to be minimized over S. For a solution $s \in S$, let $N(s)$ denote the *neighborhood* of s which is defined as the set of solutions in S obtained from s by performing a local change, called *move*. A local search is an algorithm that generates a sequence $s_0, s_1, \ldots, s_r$ of solutions in S, where s_0 is an initial solution and each s_i $(i \geq 1)$ belongs to $N(s_{i-1})$. Tabu search is one of the most famous local search algorithms. In order to avoid cycling, tabu search uses a *tabu list* T that contains forbidden moves. More precisely, a move m from s_{i-1} to s_i is forbidden, and s_i is called a *tabu solution*, if m belongs to the tabu list T and $f(s_i) \geq f(s^*)$, where s^* is the best solution encountered so far. At each iteration, the algorithm moves from the current solution s_{i-1} to the best non tabu neighbor $s_i \in N(s_{i-1})$, even if $f(s_i) > f(s_{i-1})$. The general scheme of a tabu search algorithm is given in Figure 5.7. For more details on tabu search, the reader may refer to [10].

In order to illustrate how the theoretical results of the previous section can help in the design of efficient algorithms for the ICP, we have developed two tabu search algorithms, the second one being based on Theorem 5.2.1 for avoiding the visit of equivalent solutions. Both tabu search algorithms are heuristic methods for the k-ICP. They are used to solve the ICP with the following scheme.

1. Determine an upper bound k on $\chi_{int}(G)$.
2. Apply tabu search for the $(k-1)$-ICP; if the output is a legal $(k-1)$-interval coloring then set $k \leftarrow k-1$ and repeat step 2, else return k.

Tabu search
Generate an initial solution $s \in S$, set $T \leftarrow \emptyset$ and $s^* \leftarrow s$;
while no stopping criterion is met **do**
  Determine a solution $s' \in N(s)$ with minimum value $f(s')$ such that the move from s to s' does not belong to T or $f(s') < f(s^*)$;
  if $f(s') < f(s^*)$ **then**
    set $s^* \leftarrow s'$;
  end if
  Set $s \leftarrow s'$ and update T.
end while

Figure 5.7 – General scheme of a tabu search algorithm.

Tabucol [13] is a tabu search algorithm for the classical k-coloring problem (i.e., for the k-ICP with $\omega_i = 1$ for all vertices $i \in V$). The search space S is the set of (not necessary legal) k-colorings of G. A solution $c \in S$ is therefore a partition of the vertex set into k subsets $V_1, \ldots, V_k$. The evaluation function f measures the number of conflicting edges. Hence, for a solution $c = (V_1, \ldots, V_k)$ in S, $f(c)$ is equal to $\sum_{i=1}^{k} |E_i|$, where E_i denotes the set of edges with both endpoints in V_i. The goal of *Tabucol* is to determine a k-coloring c such that $f(c) = 0$. Given a k-coloring c, a neighbor k-coloring $c' \in N(c)$ is obtained by choosing an endpoint i of a conflicting edge and assigning a new color $c'(i) \neq c(i)$ to i. When modifying the color $c(i)$ of a vertex i, the tabu list stores the ordered triple $(i, c(i), f(c))$, which means that for some number of iterations, all moves to a solution c' with $f(c') \geq f(c)$ and $c'(i) = c(i)$ have a tabu status.

The first proposed tabu search algorithm, called TABU_1, is a simple adaptation of *Tabucol* to the k-ICP. The search space S is the set of (not necessary legal) k-interval colorings of G. The evaluation function f measures the total overlap of intervals on adjacent vertices. More precisely, given a k-interval coloring I of G, we define

$$f(I) = \sum_{[i,j] \in E} |I(i) \cap I(j)|.$$

Hence, a k-interval coloring is legal if and only if $f(I) = 0$. Given a k-interval coloring I, the neighborhood of I is defined as the set of solutions I' which can be obtained from I by choosing an endpoint i of a conflicting edge and assigning a new interval $I'(i) \neq I(i)$ to i. We denote $N_1(I)$ the set of such neighbors of I. Let $\min_I(i)$ denote the smallest integer in the interval $I(i)$. When modifying the interval $I(i)$ of a vertex i, the tabu list TL_1 stores the ordered triple $(i, \min_I(i), f(I))$, which means that for some number of iterations, a move to a solution I' with $f(I') \geq f(I)$ and $\min_{I'}(i) = \min_I(i)$ (i.e., with $I(i) = I'(i)$) has a tabu status.

Consider again the *Tabucol* algorithm for the classical k-coloring problem. Let $c' \in N(c)$ be a neighbor solution of a k-coloring c obtained by modifying the color of vertex i. Then c and c' are equivalent if and only if $c(j) \notin \{c(i), c'(i)\}$ for all vertices $j \neq i$. Indeed, if there is a vertex $j \neq i$ with $c(i) = c(j)$ then i and j belong to the same color class in c but not in c'. Similarly, if $c'(i) = c(j)$ with $j \neq i$ then i and j belong to the same color class in c' but not in c. In summary, c and c' are equivalent if and only if i is the unique vertex with color $c(i)$ and the new color $c'(i)$ assigned to i is not used by any vertex $j \neq i$, which is most unlikely. The situation is totally different for the k-ICP. For example, consider the 8-interval coloring I at the top of Figure 5.8. It contains three conflicting edges, namely $[c, f]$, $[d, e]$ and $[e, f]$. Hence, a neighbor of I is obtained by changing the interval associated with vertex c, d, e or f, which gives a total of 21 neighbors in $N_1(I)$. Four of these neighbors are equivalent to I and represented at the bottom of Figure 5.8.

The second tabu search algorithm, called TABU_2, has only two differences with TABU_1. The first difference is on the definition of the neighborhood of a k-interval coloring. In order to avoid visiting equivalent solutions, the neighborhood $N_2(I)$ of a k-interval colorings I is defined as the subset of solutions $I' \in N_1(I)$ such that there exists at least one conflicting edge $[i, j]$ with $|I(i) \cap I(j)| \neq |I'(i) \cap I'(j)|$. Hence, given a k-interval coloring I, a neighbor solution $I' \in N_2(I)$ is obtained by choosing a vertex $i \in V$ that is the endpoint of a conflicting edge and assigning a new

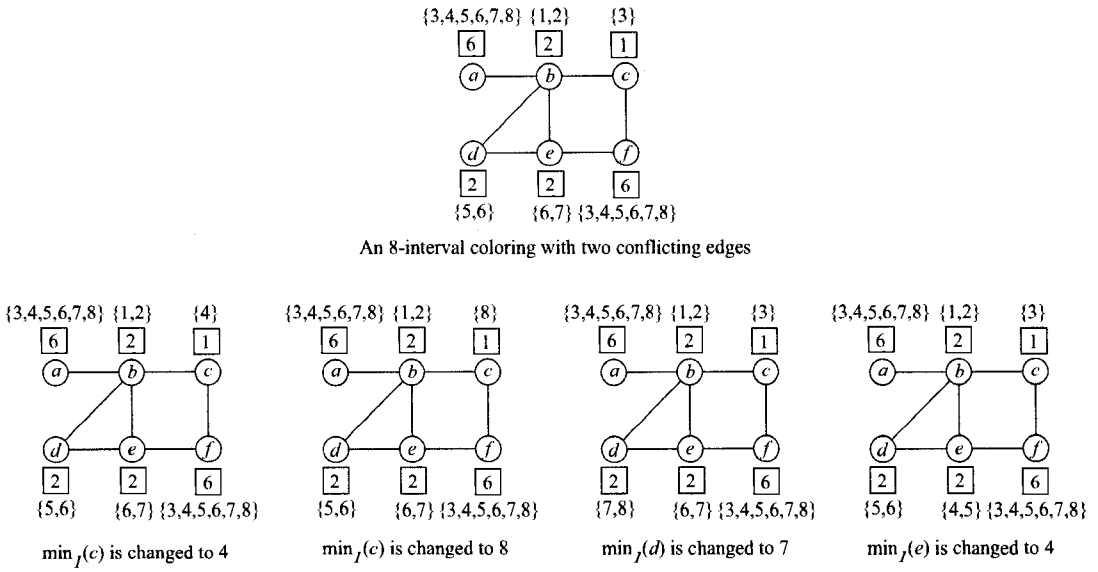


Figure 5.8 – An 8-interval coloring with 4 equivalent neighbors.

interval $I'(i) \neq I(i)$ to i such that there is at least one vertex j adjacent to i with $|I(i) \cap I(j)| > 0$ and $|I(i) \cap I(j)| \neq |I'(i) \cap I(j)|$. According to Theorem 5.2.1 this is sufficient to ensure that no solution in $N_2(I)$ is equivalent to I. For the 8-interval coloring I at the top of Figure 5.8, $N_1(I)$ contains 21 neighbors while $N_2(I)$ contains only 13 neighbors obtained by setting $\min_{I'}(c) = 1$ or 2, $\min_{I'}(d) = 1, 2, 3, 4$ or 6, $\min_{I'}(e) = 1, 2, 3, 5$ or 7, or $\min_{I'}(f) = 1$. The four neighbors $I' \in N_1(I)$ represented at the bottom of Figure 5.8 do not belong to $N_2(I)$ since they are equivalent to I. In addition, $N_2(I)$ does not contain the neighbors I' with $\min_{I'}(c) = 5, 6$ or 7 or $\min_{I'}(f) = 2$ since these values do not change the size of the overlap of the intervals on the endpoint of a conflicting edge.

The second difference between TABU_1 and TABU_2 is on the definition of a tabu move. We consider a second tabu list TL_2, and a move is declared tabu if both tabu lists assign a tabu status to the move. The second tabu list is defined as follows. When modifying the interval associated with a vertex i for moving from a solution I to a

neighbor solution $I' \in N_2(I)$, we consider all vertices j such that $|I(i) \cap I(j)| > 0$ and $|I(i) \cap I(j)| \neq |I'(i) \cap I(j)|$, and for each such vertex we insert the ordered quadruple $(i, j, |I(i) \cap I(j)|, f(I))$ in a second tabu list TL_2. The move from a solution I to a solution I' is considered as tabu according to TL_2 if there exists $(i, j, q, r) \in TL_2$ such that $|I(i) \cap I(j)| \neq |I'(i) \cap I'(j)| = q$ and $f(I') \geq r$. Notice that an ordered quadruple (i, j, q, r) is introduced in TL_2 only if $q > 0$, which means that we never impose that two intervals should not overlap since this would forbid the visit of too many solutions. For illustration, if we move from the 8-interval coloring I at the top of Figure 5.8 to a neighbor solution I' by setting $I'(f) = \{1, 2, 3, 4, 5, 6\}$, then $(a, f, 6, 4)$ and $(e, f, 2, 4)$ are introduced in the tabu list. While $|I(b) \cap I(f)| \neq |I'(b) \cap I'(f)|$, the ordered quadruple $(b, f, 0, 4)$ does not enter the tabu list since $|I(b) \cap I(f)| = 0$.

5.3.2 Computational experiments

We first report computational experiments on 12 DIMACS benchmark graphs having up to 125 vertices. These instances have also been considered in [1], and therefore constitute a test set on which comparisons can be made with other algorithms. For a detailed description of these instances, the reader can refer to [19].

For measuring the performance of the proposed algorithms, we also report known lower and upper bounds on the optimal solution. More precisely, Čangalović and Schreuder in [4] have described an exact algorithm for finding the interval chromatic number. It is based on the Branch-and-Bound principle. An initial lower bound $LB(G)$ on $\chi_{int}(G)$ is obtained by determining a clique of maximum total weight, using a variation of the algorithm proposed in [18]. Also, an initial upper bound $UB(G)$ on $\chi_{int}(G)$ is obtained by using the heuristic algorithm proposed in [3]. Moreover, two truncated Branch-and-Bound algorithms for the ICP are proposed in [1]. Both algorithms are run with a time limit of one hour. Their output is either the optimal value (i.e., the interval chromatic number), or an upper bound on $\chi_{int}(G)$.

Table 5.1 – Results for DIMACS benchmark graphs.

instance							$TABU_1$			$TABU_2$		
name	n	m	max ω	LB	UB	Trunc BB	Best	Worse	Average	Best	Worse	Average
DSJC125.1g	125	736	5	19	23	**19**	**19**	**19**	**19**	**19**	**19**	**19**
DSJC125.5g	125	3891	5	40	72	68	62	63	62.8	62	63	62.8
DSJC125.9g	125	6961	5	122	166	163	153	154	153.4	152	154	152.8
R50_5g	50	612	5	27	34	32	31	32	31.6	31	32	31.4
R50_9g	50	1092	5	64	73	68	66	66	66	66	66	66
R75_1g	70	251	5	14	19	**16**	**16**	**16**	**16**	**16**	**16**	**16**
R75_5g	75	1407	5	31	50	46	43	44	43.2	43	44	43.2
R75_9g	75	2513	5	85	104	101	95	96	95.4	96	96	96
R100_1g	100	509	5	15	19	**17**	**17**	**17**	**17**	**17**	**17**	**17**
R100_5g	100	2456	5	35	58	54	49	50	49.8	49	50	49.6
R100_9g	100	4438	5	108	140	134	126	127	126.8	125	127	126.4
average				50.91	68.91	65.27	61.54	62.18	61.90	61.45	62.18	61.83

When using $TABU_1$ or $TABU_2$, we start the search with $k = UB(G)$. Then, as explained at the beginning of Section 5.3, we decrease k by one unit if a legal $(k-1)$-interval coloring is found, and this process is repeated until a time limit of 20 minutes is reached. All tests were performed on an Intel(R) Core(TM)2 cpu 6400/2.13GHz. According to preliminary experiments, the duration of a tabu status for the first tabu list is randomly chosen at each iteration in the interval $[\sqrt{k|V|}, 3\sqrt{k|V|}\,]$ while the interval $[\frac{1}{2}|V|\sqrt{\max_{i\in V}\omega_i}\,,\ \frac{3}{2}|V|\sqrt{\max_{i\in V}\omega_i}\,]$ is used for the second tabu list. A better tuning of these parameters is certainly possible, but the chosen values turned to be the best for our limited test set.

The three first columns of Table 5.1 contain the name of the instances, their number n of vertices, and their number m of edges. The next column indicates the largest vertex weight (column labeled "max ω") which also corresponds to the number of different vertex weights. We then report the value of the lower and upper bounds $LB(G)$ and $UB(G)$ (columns labeled "LB" and "UB") mentioned above. Column labeled "Trunc BB" contains the best upper bound obtained in [1] with one of the two truncated Branch-and-Bound algorithms. When a proof of optimality was obtained, we use bold numbers. The next three columns contain the results obtained using $TABU_1$.

We ran $TABU_1$ five times on each graph, and columns "Best", "Worse" and "Average" contain the best, the worse and the average solution values we have reached. Again, we use bold numbers when a solution produced with $TABU_1$ is known to be optimal (because it reaches the lower bound $LB(G)$ or is equal to an optimal value reported in column "Trunc BB"). The last three columns contain the same information for $TABU_2$. The last line contains average numbers for each column.

We observe that both algorithms produce better results than those reported in [1] (column "Trunc BB"). We cannot conclude from these 12 benchmark problems that $TABU_2$ is better than $TABU_1$. However, we can observe that the use of Theorem 5.2.1 for avoiding the visit of equivalent solutions has helped to reduce the number of colors to 152 for instance DSJC125.9g and to 125 for R100_9g. Although $TABU_2$ is in average slightly better than $TABU_1$, it could never reach a 95-interval coloring for R75_9g, while such a solution was obtained on three of the five runs with $TABU_1$.

While experimenting $TABU_1$ and $TABU_2$ on other instances, we have noticed that $TABU_2$ tends to produce better results than $TABU_1$ only on instances in which the vertices have almost all the same weights. To illustrate this fact, we now report results obtained on a second set of instances and then give a possible explanation of this observation.

Given a positive integer n, a real number $p \in [0, 1]$ and a set Ω of positive integers, a random graph $G_{n,p,\Omega}$ contains n vertices, all $n(n-1)/2$ ordered pairs of vertices have a probability p of being linked by an edge, and the weight ω_i of a vertex i is chosen randomly according to a uniform distribution in Ω. We have generated such random graphs with $n = 100, 125$ and 250, $p = 0.5$, and with three different weight sets $\Omega_1 = \{7, 9\}$, $\Omega_2 = \{5, 7, 9, 11\}$ and $\Omega_3 = \{1, 3, 5, 7, 9, 11, 13, 15\}$. For each triplet (n, p, Ω_i), four graphs were created and we have then run $TABU_1$ and $TABU_2$ five times on each graph. Average results are reported in Figure 5.9. More precisely, for

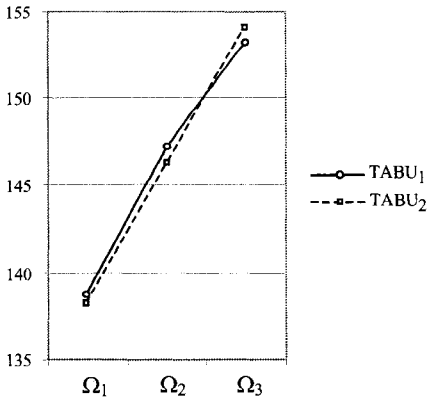


Figure 5.9 – Experiments on random graphs with various weight sets.

each set Ω_i we give the average number of colors used by TABU_1 and TABU_2 on the sixty runs (five runs on each of the twelve graphs), the plain line being for TABU_1 and the dotted one for TABU_2. We observe that TABU_2 is better than TABU_1 with the weight sets Ω_1 and Ω_2, and worse with Ω_3 which contains many different weights.

Given a k-interval coloring I, the solutions I' in $N_2(I)$ are obtained from I by assigning a new interval $I'(i)$ to an endpoint i of a conflicting edge so that $|I(i) \cap I(j)| \neq |I'(i) \cap I'(j)|$ for at least one conflicting edge $[i,j]$. As observed in Section 5.3.1, this is a sufficient condition to ensure that there is no solution equivalent to I in $N_2(I)$. The condition is however not necessary since it may happen that $|I(i) \cap I(j)| = |I'(i) \cap I'(j)|$ for all conflicting edges $[i,j]$ while the existence of a vertex j' with $|I(i) \cap I(j')| \neq |I'(i) \cap I'(j')|$ makes I' not equivalent to I. In summary, it may happen that many solutions in $N_1(I)$ but not in $N_2(I)$ are not equivalent to I and we now show that this turns to be particularly true when the weights on the vertices have many different values.

Consider any conflicting edge $[i,j]$ and assume first that $I(i) \not\subset I(j)$ and $I(j) \not\subset I(i)$. Then there is only one change of $I(i)$ to $I'(i)$ which gives $|I(i) \cap I(j)| = |I'(i) \cap I'(j)|$ (if $\min_I(i) \neq \min_I(j)$). For example, if $I(i) = \{1,2,3\}$ and $I(j) = \{3,4,5,6,7,8\}$,

then $|I(i) \cap I(j)| = |I'(i) \cap I'(j)|$ only if $I'(i) = \{8, 9, 10\}$. The situation is different when $I(i) \subset I(j)$ or $I(j) \subset I(i)$. Indeed, there are then $|\omega_i - \omega_j|$ possible ways of changing $I(i)$ to $I'(i)$ so that $|I(i) \cap I(j)| = |I'(i) \cap I'(j)|$. If $[i, j]$ is the unique conflicting edge, then none of these solutions belong to $N_2(I)$ while there all belong to $N_1(I)$ and are possibly all non equivalent to I. For example, if $I(i) = \{4, 5, 6\}$ and $I(j) = \{3, 4, 5, 6, 7, 8\}$, then the solutions obtained by setting $\min_I(i) = 3, 5$ or 6 do not belong to $N_2(I)$ while the existence of a vertex j' not adjacent to i with $I(j') = \{6, 7, 8\}$ is sufficient to prove that I' is not equivalent to I if $\min_I(i)$ is set equal to 3,5 or 6.

In summary, the number of solutions I' in $N_1(I)$ not equivalent to I and not in $N_2(I)$ is proportional to the values $|\omega_i - \omega_j|$ of the conflicting edges $[i, j]$. Such values are small only when the weights on the vertices are almost all the same, which gives a possible explanation of the curves in Figure 5.9.

A better neighborhood for tabu search would contain all solutions in $N_1(I)$ which are not equivalent to I, but this would increase the computational complexity. Indeed, for the endpoint i of a conflicting edge, one can test in $O(|V|)$ whether a change of $I(i)$ to a new interval $I'(i)$ gives a solution I' equivalent to I. Hence, such a neighborhood could be generated with a time complexity in $O(F(I)|V|k)$, where $F(I)$ denotes the number of endpoints of conflicting edges in a k-interval coloring I. While $F(I) \in O(|V|)$ (which gives a theoretical time complexity in $O(|V|^2 k)$, $F(I)$ typically contains only few vertices which gives a time complexity in $O(|V|k)$ in practice. For comparison, the generation of $N_2(I)$ can be performed in time complexity in $O(F'(I)k)$, where $F'(I)$ denotes the number of conflicting edges in I. This is theoretically in $O(|E|k)$ but typically in $O(k)$ since the number of conflicting edges in $TABU_2$ reduces very quickly to a few units.

5.4 Conclusion

Two k-interval colorings I_1 and I_2 are said *equivalent* if there is a permutation π of the integers $1, \cdots, k$ such that $\ell \in I_1(i)$ if and only if $\pi(\ell) \in I_2(i)$ for all vertices $i \in V$. We have shown that a necessary and sufficient condition for such an equivalence is to have $|I_1(i) \cap I_1(j)| = |I_2(i) \cap I_2(j)|$ for all vertices i and j. Hence, equivalent solutions to the k-ICP are easy to recognize and we have shown that a tabu search algorithm for the k-ICP can possibly be improved by forbidding the visit of equivalent solutions.

While the two proposed tabu search algorithms produce reasonably good results in comparisons with those published in [1], we do not argue that they constitute the best possible algorithms for the ICP. The experiments reported in Section 5.3.2 should help to orient future research on the development of more elaborate algorithms for the ICP.

BIBLIOGRAPHIE

[1] Bouchard, M., Čangalović, M., Hertz, A. (2007). On a reduction of the interval coloring problem to a series of bandwidth coloring problems. *Technical report G-2007-69, Les Cahiers du GERAD*, Montréal, Canada.

[2] Buchsbaum, A.L., Karloff, H., Kenyon, C., Reingold, N., Thorup, M. (2003). OPT versus LOAD in dynamic storage allocation. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 632–646.

[3] Clementson, A.T., Elphick, C.H. (1983). Approximate coloring algorithms for composite graphs *Journal of Operational Research Society*, 34/6 :503–509.

[4] Čangalović, M., Schreuder, J.A.M. (1991). Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths. *European Journal of Operational Research*, 51 :248–258.

[5] Fabri, J. (1979). Automatic storage optimization. *ACM SIGPLAN Notices : Proceedings of the 1979 SIGPLAN symposium on Compiler construction*, 14/8 :83–91.

[6] Galinier, P., Hao, J.K. (1999). Hybrid evolutionary algorithms for graph colorings. *Journal of Combinatorial Optimization*, 3 :379–397.

[7] Galinier, P., Hertz, A. (2006). A survey of local search methods for graph coloring. *Computers & Operations Research*, 33 :2547-2562.

[8] Garey, M.R., Johnson, D.S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W.H. Freman and Company, NY.

[9] Gergov, J. (1999). Algorithms for compile-time memory optimization. *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, 907–908.

[10] Glover, F., Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, Boston.

[11] Golumbic, M. (1980). *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, NY.

[12] Hajós, G. (1957). Über eine Art von Graphen. *Int. Math. Nachrichtern*, 11.

[13] Hertz, A., de Werra, D. (1987). *Using Tabu Search Techniques for Graph Coloring*, Computing 39, 345-351.

[14] Kubale, M. (1989). Interval vertex-coloring of a graph with forbidden colors. *Discrete Mathematics*, 74 :125–136.

[15] Pardalos, P.M., Mavridou, T., Xue, J. (1998). The graph coloring problem : A bibliographic survey. *Handbook of Combinatorial Optimization.* Kluwer Academic Publishers, 2 :331–395.

[16] Pemmaraju, S.V., Penumatcha, S., Raman, R. (2005). Approximating Interval Coloring and Max-Coloring in Chordal Graphs *ACM Journal of Experimental Algorithmics*, 10 :1–19.

[17] Punter, A. (1976). *Systems for timetabling by computer based on graph coloring.* Ph.D. Thesis, C.N.A.A., Hatfield Polytechnic.

[18] Sakaki, T., Nakashima, K., Hattori, Y. (1976). Algorithms for finding in the lump both bounds of the chromatic number of a graph. *The Computer Journal*, 19 :329-332.

[19] Trick, M.A. (2002). Computational symposium : Graph coloring and its generalizations. Cornell University, Ithaca, NY. `http ://mat.gsia.cmu.edu/COLOR04`.

[20] de Werra, D., Hertz, A. (1988). Consecutive colorings of graphs. *Zeitschrift für Operations Research*, 32/1 :1-8.

CHAPITRE 6 : UNE APPROCHE DE PROGRAMMATION EN NOMBRES ENTIERS POUR LA RÉSOLUTION D'UN PROBLÈME D'HORAIRE

6.1 Introduction

Les problèmes de planification d'horaires sont nombreux et très diversifiés et les approches de résolution varient énormément selon le type de problèmes rencontrés. Dans ce chapitre, nous considérons un cas particulier de ces problèmes qui consiste à attribuer des activités, de diverses natures à des quarts de travail fixés, i.e leurs heures de début et de fin et leurs heures de pauses sont connues et fixées. À notre connaissance les approches utilisées pour ce genre de problème font toutes appel à des méthodes de résolution de programmes linéaires en nombres entiers (PLNE) utilisant la génération de colonnes. L'approche de résolution que nous proposons, quoi que reposant elle aussi sur un PLNE, utilise un nouveau type de modèle basé sur deux problèmes de flot dans un réseau. Le modèle proposé en plus de son efficacité a l'avantage de pouvoir être facilement mis en application en utilisant des outils génériques de résolution pour le PLNE.

Dans la première section de ce chapitre, nous présentons une description du problème que nous proposons de résoudre. La section suivante expose une revue des travaux présentés dans la littérature concernant des problèmes similaires à celui que nous étudions. La troisième section présente un algorithme heuristique basé sur la méthode tabou ainsi qu'une approche permettant de réduire la taille du problème. Nous poursuivons ensuite en présentant notre modèle et ces différentes variantes, lesquels

sont une extension des approches utilisées pour réduire la taille du problème. La cinquième section est consacrée à la présentation des expérimentations numériques que nous avons menées et à la discussion des résultats obtenus. Finalement, nous terminons par une brève conclusion.

6.2 Description du problème

La plupart des problèmes de planification d'horaires vise à définir des quarts de travail pour une période donnée et à assigner ces quarts à des employés. Le problème consiste alors à choisir parmi un vaste ensemble de types de quarts possibles ceux qui permettront de répondre au mieux les objectifs fixés tout en respectant un certain nombre de contraintes. Ce que l'on retrouve généralement dans la littérature est le cas où un seul type de travail (activité) est effectué à l'intérieur d'un même quart de travail : on parle alors d'un contexte mono-activité. Dans le cas plus rarement étudié où plusieurs types d'activités sont possibles à l'intérieur d'un même quart de travail, on parlera de contexte multi-activités. Dans un contexte multi-activités, le nombre de quarts possibles devient souvent très grand et il est commun de séparer le problème en deux phases : la première détermine le début et la fin des quarts, les périodes de pause et l'employé qui effectuera le quart de travail ; la deuxième phase consiste à assigner les activités qui rempliront les quarts fixés à la première phase. Le problème étudié dans ce chapitre est un cas assez typique de ceux qui doivent être résolus lors de cette deuxième phase. Dans cette section, nous décrivons en détail le problème auquel nous nous sommes attaqué et nous présentons ensuite quelques définitions qui nous serviront au fil des sections suivantes.

6.2.1 Le contexte multi-activités avec quarts de travail fixés

Le problème d'attribution d'activités à des quarts de travail vise à assigner un certain nombre de tâches à des quarts pré-déterminés. Un horizon de planification est découpé en tranches de temps de durée égale que nous appelons périodes. Pour une période t donnée, on trouve, pour chaque activité a, une demande de couverture δ_{at}. Un quart est divisé en pièces de travail qui sont des séquences de périodes de travail sans pause, les heures de début et de fin de ces pièces de travail sont connues et fixées. Chaque quart de travail doit être comblé par des blocs d'activité (i.e. une suite de périodes consécutives où une seule activité est présente). Les blocs d'activité doivent durer un minimum de temps $d_{min,a,t}$ et ne doivent pas dépasser une certaine durée $d_{max,a,t}$. Ces deux valeurs dépendent du type d'activité et de la période à laquelle elle débute. De plus, à chaque quart q est associé un ensemble d'activités A_q permises durant le quart, qui correspond aux qualifications de l'employé affecté au quart. On dit qu'il y a une transition quand, à l'intérieur d'une même pièce de travail, on passe de l'activité a pour la période t à une autre activité $a' \neq a$ à la période $t+1$. L'objectif est de remplir les quarts de sorte à minimiser la sous-couverture, la sur-couverture et aussi de minimiser le nombre de transitions. Une sous-couverture (resp. une sur-couverture) est le manque (resp. surplus) d'une unité de demande pour une activité durant une période. Plus précisément, soit $z(x)$ l'objectif de la solution x, r le nombre de transitions et x_{aqt} une variable valant 1 si l'activité a est assignée au quart q à la période t et 0 sinon. Si A est l'ensemble des activités, Q l'ensemble des quarts, T l'ensemble des périodes de l'horizon et T_q l'ensemble des périodes appartenant au quart $q \in Q$, on a :

$$z = c_t \cdot r + \sum_{t \in T} \sum_{a \in A} (c_{uc} \cdot \max\{0, \delta_{at} - \sum_{q \in Q | t \in T_q} x_{aqt}\} + c_{oc} \cdot \max\{0, \sum_{q \in Q | t \in T_q} x_{aqt} - \delta_{at}\}) \tag{6.1}$$

où c_{uc} est le coût associé à une sous-couverture, c_{oc} le coût associé à une sur-couverture et c_t est la pénalité imposée pour chaque transition d'une activité à une autre à l'intérieur d'une même pièce de travail dans un quart.

6.2.2 Définitions

Nous introduisons ici un certain nombre de définitions et quelques variables de décision. Celles-ci serviront à exposer les différentes méthodes élaborées pour résoudre le problème ainsi qu'à présenter les différentes approches que l'on retrouve dans la littérature dans un contexte unifié.

- P, l'ensemble des pièces de travail ;
- Q, l'ensemble des quarts de travail ;
- A, l'ensemble des activités ;
- A_q, l'ensemble des activités permises durant le quart $q \in Q$;
- T, l'ensemble des périodes ;
- B, l'ensemble des blocs d'activité respectant les durées minimale et maximale ;
- S_{pat} l'ensemble des blocs de la pièce p couvrant l'activité a et commençant à la période t ;
- E_{pat} l'ensemble des blocs de la pièce p couvrant l'activité a et terminant à la période t ;
- δ_{at}, demande pour l'activité a à la période $t \in T$;
- x_b, variable de décision valant 1 si le bloc b appartient à la solution, 0 sinon ;
- u_{at}, variable entière correspondant au nombre de sous-couvertures pour l'activité a à la période t ;
- o_{at}, variable entière correspondant au nombre de sur-couvertures pour l'activité a à la période t ;
- c_{uc}, coût pour une sous-couverture ;

- c_{oc}, coût pour une sur-couverture ;
- c_t, coût pour une transition.

Notons que selon la description faite du problème d'attribution d'activités à des quarts de travail, les ensembles B, S_{pat} et E_{pat} sont déterminés par les durées minimales et maximales $d_{min,a,t}$ et $d_{max,a,t}$. Cependant, pour le reste du chapitre, nous utiliserons uniquement les ensembles B, S_{pat} et E_{pat} et les méthodes présentées pourraient tout aussi bien s'appliquer à d'autres types de restrictions sur l'ensemble des blocs permis.

6.3 Revue de littérature

Dans cette section, nous faisons une revue des travaux pertinents portant sur deux types de problèmes de planification d'horaires et sur un problème d'ordonnancement dont l'évolution des approches de résolution présente des similitudes avec notre démarche. Le premier type de problème considéré porte sur la construction de quarts de travail. Le deuxième type étudié est celui qui a été présenté à la section précédente. Finalement, nous présentons un certain nombre de travaux portant sur le problème visant à minimiser la somme pondérée des retards lors de l'ordonnancement des tâches sur une machine, lesquels présentent des caractéristiques de modélisation similaires aux modèles que nous présenterons à la quatrième section de ce chapitre.

6.3.1 Construction de quarts de travail

Les premiers travaux sur la construction de quarts de travail ont été présentés par Dantzig (1954) et portaient sur des problèmes précédemment considérés par Edie (1954) qui sont de nature mono-activité. Dantzig a présenté une formulation basée

sur un modèle de recouvrement d'ensemble généralisé dans lequel chaque quart possible est représenté explicitement par une variable de décision. Plus précisément, on a Ω l'ensemble des types de quarts possibles et à chaque élément ω de cet ensemble est associé une variable entière Θ_ω indiquant combien de quarts possédant les caractéristiques de ω doivent se retrouver dans la solution. De plus, on définit un coefficient $\alpha_{\omega,t}$ indiquant si un travail est accompli par le type de quart $\omega \in \Omega$ à la période $t \in T$. En représentant par δ_t le nombre d'unités de travail requis à la période t et en considérant que l'objectif du problème est de minimiser le nombre de quarts, Dantzig a proposé la formulation mathématique suivante :

$$\min \sum_{\omega \in \Omega} \Theta_\omega \tag{6.1}$$

sujet à :

$$\sum_{\omega \in \Omega} \alpha_{\omega,t} \Theta_\omega \geq \delta_t, \ \forall t \in T \tag{6.2}$$

$$\Theta_\omega \text{ entier}, \forall \omega \in \Omega. \tag{6.3}$$

Ce modèle a le mérite d'être simple tout en permettant une grande souplesse sur le type de quarts possibles. De plus, le modèle peut s'étendre à de nombreux contextes, notamment, comme nous le verrons, au cas multi-activités. Le nombre de variables nécessaires est généralement très élevé et le devient d'autant plus que les paramètres définissant un quart sont nombreux. C'est le cas par exemple si ceux-ci peuvent être composés de différents types d'activités. Cette approche à servi de base à plusieurs travaux subséquents.

Une approche alternative à l'énumération explicite des quarts de travail est d'utiliser des variables et des contraintes pour modéliser implicitement certaines caractéristiques du quart, le but cherché étant de diminuer la taille des problèmes. Une première approche dans ce sens à été proposée par Moondra (1976) alors qu'il travaillait

sur des problèmes où certains quarts avaient une flexibilité quant à leur début et leur longueur. Cette flexibilité a été modélisé à l'aide de variables définissant les temps de début et de fin de quart et un ensemble de contraintes forçant le respect des durées maximale et minimale de ces quarts.

Bechtold et Jacobs (1990) ont proposé une approche différente permettant de réduire le nombre de variables en représentant implicitement le placement des pauses. Dans les problèmes qu'ils ont considérés, les types de quarts sont définis par trois paramètres : l'heure de début du quart, la durée du quart et l'heure de début de la pause unique de durée fixe. Leur technique permet d'éliminer le temps de début de la pause des paramètres à considérer lors de l'énumération des variables Θ_ω représentant le nombre de quarts d'un certain type à utiliser. Si certaines conditions sont respectées, le placemement de la pause peut être effectué à l'aide de variables indiquant combien de pauses commencent à chaque période de l'horizon de planification. Ces variables sont liées aux variables de type de quart à l'aide de contraintes spéciales dites *en avant* et *en arrière.*

Différentes approches de représentation implicite ont aussi été proposées par Thompson (1995), Aykin (1996, 1998, 2000) et Rekik *et al.* (2004). Ces différents travaux ont permis d'étendre la gamme de problèmes pouvant être modélisés à l'aide d'une représentation implicite.

6.3.2 Le problème d'attribution d'activités à des quarts de travail

Les approches connues de résolution pour ce type de problème considèrent le problème présenté à la section précédente comme la structure principale des différentes variantes du problème qu'elles ont à résoudre. Elles utilisent, pour l'essentiel, le même

modèle de base auquel viennent se greffer différentes modifications et ajouts permettant d'étendre l'ensemble de variantes du problème pouvant être résolues. Ce modèle de base représente justement une partie commune de ces variantes qui correspond au problème d'attribution d'activités à des quarts de travail tel que nous l'avons défini. Le souci de présenter un modèle de base souple pouvant facilement intégrer l'ajout de nouvelles contraintes a orienté la recherche dans ce domaine vers des approches basées sur la génération de colonnes, ainsi un grand nombre de contraintes complexes peuvent être transférées au niveau des sous-problèmes.

Omari (2002) a proposé un algorithme heuristique permettant de résoudre le problème d'attribution d'activités à des quarts sur un horizon continu pour les contrôleurs aériens. Suivant l'approche de recouvrement d'ensembles généralisé proposée par Dantzig, on définit Ω_q l'ensemble des assignations possibles d'activités pour le quart fixé $q \in Q$. Un quart auquel on ajoute une attribution d'activités pour toutes ses périodes est un quart rempli. À chaque élément ω de cet ensemble est associé une variable de décision Θ_ω indiquant si le quart rempli ω est présent dans la solution. De plus, un coût c_ω est associé à ce quart rempli. On définit aussi un coefficient $\alpha_{\omega at}$ indiquant si l'activité $a \in A$ est couverte par le quart rempli $\omega \in \Omega$ à la période $t \in T$. Pour ce problème, on impose une pénalité si la demande n'est pas atteinte ou si elle est excédée. Omari propose donc la formulation mathématique suivante :

$$\min \sum_{q \in Q} \sum_{\omega \in \Omega_q} c_\omega \Theta_\omega + \sum_{a \in A} \sum_{t \in T} c_{uc} u_{at} + \sum_{a \in A} \sum_{t \in T} c_{oc} o_{at} \tag{6.4}$$

sujet à :

$$\sum_{q \in Q} \sum_{\omega \in \Omega_q} \alpha_{\omega at} \Theta_\omega + u_{at} - o_{at} = \delta_{at}, \ \forall a \in A, \forall t \in T \tag{6.5}$$

$$\sum_{\omega \in \Omega_q} \Theta_\omega = 1, \ \forall q \in Q \tag{6.6}$$

$$\Theta_\omega \in \{0,1\}, \forall \omega \in \Omega \tag{6.7}$$

$$u_{at}, o_{at} \text{ entier}, \forall a \in A, \forall t \in T. \tag{6.8}$$

Le problème consistant à résoudre la relaxation linéaire de ce modèle est appelé le problème maître, dans le cas où on n'utilise qu'un sous-ensemble des variables Θ_ω on parle de problème maître restreint. Une approche de génération de colonnes, imbriquée dans une méthode de séparation et d'évaluation progressive, est utilisée pour trouver des solutions entières et contourner la difficulté associée au grand nombre de variables Θ_ω. Ainsi, pour chaque quart $q \in Q$, un problème de plus court chemin est utilisé pour générer une nouvelle colonne, i.e. une variable Θ_ω de l'ensemble Ω_q de plus petit coût réduit. Certaines arêtes de ce graphe modélisent une transition et ont un coût de c_t, d'autres modélisent l'assignation d'une activité à une ou plusieurs périodes du quart q. Afin que le problème de plus court chemin revienne à trouver la colonne de plus petit coût réduit, chaque arête e représentant une assignation se voit attribuer un coût déterminé par les valeurs des variables duales qui correspondent aux contraintes (6.5).

Lors de la résolution de la relaxation linéaire d'un nœud de l'arbre de branchement, si toutes les colonnes présentement introduites dans le problème maître ont un coût réduit positif ou nul, alors les sous-problèmes de plus court chemin sont

résolus afin de générer de nouvelles colonnes ayant un coût réduit strictement négatif. Si, à ce moment, tous les sous-problèmes génèrent des solutions non négatives, alors la relaxation linéaire du nœud est optimale. Dans ces travaux, Omari utilise deux resources pour modéliser les contraintes de durée maximale, les sous-problèmes résultant sont des problèmes de plus court chemin avec ressources (voir Irnich et Desaulniers 2005). Notons qu'à l'exception des contraintes modélisées avec des ressources, ces sous-problèmes peuvent se formuler comme un programme linéaire et que ces formulations peuvent être combinées et ajoutées à une version modifiée du problème maître pour obtenir un PLNE qui est une formulation implicite du problème d'attribution d'activités à des quarts de travail. Une formulation implicite complète pourrait être obtenue en ajoutant des contraintes pour faire respecter les durées maximales d'affectation à une même activité.

Au delà de ce modèle de base, Omari ajoute plusieurs autres contraintes dont nous ne parlerons pas. Une heuristique de décomposition temporelle est aussi utilisée pour découper les problèmes en une suite de problèmes plus petits. Se basant sur les travaux d'Omari, Vatri (2001) et Bouchard (2004) ont généralisé cette technique pour permette d'établir, pour chaque jour de travail de l'employé, l'heure de début et de fin de son quart de travail pour cette journée, de définir les périodes de pauses et d'assigner des activités à ces quarts.

6.3.3 Ordonnancement sur une machine

On retrouve dans Bigras (2008) une présentation de l'évolution des modèles de PLNE qui met en évidence un développement similaire à celui que nous proposons dans ce chapitre. Ce développement consiste à reconnaître qu'un certain nombre de contraintes dans la formulation correspondent à un problème de flot et ensuite à remplacer ces contraintes par les contraintes du problème de flot correspondant afin

de diminuer le nombre d'éléments non nuls dans la matrice. Le problème considéré est celui de minimiser la somme pondérée des retards pour l'ordonnancement d'une machine $(1||\sum w_j T_j)$, en particulier le cas, similaire au problème qui nous intéresse, où l'horizon de planification est discrétisé en T périodes. Dans ce contexte on peut définir les variables binaires x_{jt} valant 1 si la tâche j commence à la période t et 0 sinon et on pose c_{jt} le coût associé à cette variable. Le problème comporte n tâches, chacune ayant une durée p_j, et la période t commence à $t-1$ et se termine à t. En utilisant ces définitions, on peut présenter le modèle proposé par Dyer et Wolsey (1990) de la manière suivante :

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} c_{jt} x_{jt} \tag{6.9}$$

sujet à :

$$\sum_{t=1}^{T-p_j+1} x_{jt} = 1, \ \ j = 1, \ldots, n \tag{6.10}$$

$$\sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} x_{js} \leq 1, \ \ t = 1, \ldots, T \tag{6.11}$$

$$x_{jt} \in \{0,1\}, \ \ j = 1, \ldots, n, t = 1, \ldots, T - p_j + 1. \tag{6.12}$$

Nemhauser and Wolsey (1988) ont observé que si l'on relaxe les contraintes (6.10), le problème résultant est un problème de plus court chemin dans un réseau. Le modèle suivant est une formulation équivalente au modèle précédent mais ayant une matrice moins dense. Cette dernière est obtenue en remplaçant les contraintes (6.11) par les contraintes de conservation de flot du problème de plus court chemin correspondant :

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T-p_j+1} c_{jt} x_{jt} \tag{6.13}$$

sujet à :

$$\sum_{t=1}^{T-p_j+1} x_{jt} = 1, \ j = 1, \ldots, n \tag{6.14}$$

$$\sum_{j=1}^{n} x_{j1} + e_1 \leq 1, \tag{6.15}$$

$$\sum_{j=1}^{n} x_{j,t-p_j} + e_{t-1} - \sum_{j=1}^{n} x_{jt} - e_t = 0, \ t = 2, \ldots, T \tag{6.16}$$

$$\sum_{j=1}^{n} x_{j,T-p_j+1} + e_T \leq 1, \tag{6.17}$$

$$x_{jt} \in \{0,1\}, \ j = 1, \ldots, n, t = 1, \ldots, T - p_j + 1. \tag{6.18}$$

où e_t est une variable de décision qui modélise un temps mort sur la machine pour la période t. Ces variables sont nécessaires pour conserver l'équivalence entre les deux modèles mais pourraient être enlevées puisqu'il existe toujours une solution optimale sans temps mort. C'est cette idée de remplacer certaines contraintes par des contraintes de conservation de flot afin de réduire la densité de la matrice qui sera repris dans ce chapitre dans le but d'accélérer la résolution des modèles de PLNE proposés.

6.4 Algorithme basé sur la méthode tabou

Dans cette section, nous présentons un algorithme basé sur la méthode de recherche tabou (voir Glover et Laguna 1997) pour résoudre une version simplifiée du problème

d'assignation d'activités à des quarts de travail. Le problème est simplifié en supposant qu'une solution sans sous/sur-couverture existe. Nous présentons aussi quelques méthodes permettant de réduire la taille du problème.

6.4.1 Aperçu de la méthode

L'algorithme proposé est basé sur une méthode tabou. L'espace des solutions est composé de deux classes de solutions distinctes. D'abord S_1 l'ensemble des attributions d'activités respectant les durées minimales et maximales d'affectation et S_2 l'ensemble des assignations d'activités pouvant violer les durées minimales et maximales d'affectation mais ne générant aucune sous-couverture (en réalité certaines sous-couvertures, dites inévitables, seront permises, nous en parlerons plus loin) et aucune sur-couverture. Une solution de la méthode est un point dans $S_1 \times S_2$, c'est-à-dire la combinaison d'une solution dans S_1 et d'une solution dans S_2. Une solution x (resp. y) est exprimée par les variable x_{aqt} (resp. y_{aqt}) valant 1 si l'activité a est assignée au quart q à la période t et 0 sinon. Le coût $z'(x, y)$ d'une telle paire $(x, y) \in S_1 \times S_2$ est :

$$z'(x,y) = c_t \cdot r + \sum_{q \in Q} \sum_{t \in T_q} \sum_{a \in A} |x_{aqt} - y_{aqt}|(c_{oc} + c_{uc}) \tag{6.1}$$

où r est le nombre de transition dans la solution x. La valeur de $z'(x, y)$ n'est pas identique à $z(x)$ mais $z'(x, y)$ est une borne supérieure sur $z(x)$ lorsque $S_2 \neq \emptyset$. On peut se convaincre de cela en constatant que si, à la période t dans la solution x, il y a k sur-couvertures pour l'activité a alors puisque $S_2 \neq \emptyset$ ces k sur-couvertures génèrent k sous-couvertures sur certaines des autres activités. De plus, puisque $k' = \sum_{q \in Q} |x_{aqt} - y_{aqt}| \geq \sum_{q \in Q}(x_{aqt} - y_{aqt}) = k$, ces k sur-couvertures et sous-couvertures

qui ont un coût de $k(c_{uc} + c_{oc})$ dans $z(x)$ génèrent un coût plus grand ou égal de $k'(c_{uc} + c_{oc})$ dans $z'(x, y)$. Il en est ainsi pour toutes les autres sur-couvertures et pour toutes les périodes. Notons aussi que z et z' ne sont pas toujours identiques, soit x, y deux solutions réalisables ayant r transitions telle que $z(x) = z(y) = rc_t$ (pas de sur-couverture et de sous-couverture), si l'assignation pour la période t est $(q_1, a_1), (q_2, a_2)$ pour x et $(q_1, a_2), (q_2, a_1)$ pour y alors $z'(x, y) \geq z(x) + 2(c_{oc} + c_{uc})$.

L'algorithme 6.1 décrit l'algorithme de recherche tabou que nous avons développé pour résoudre le problème d'attribution d'activités à des quarts de travail.

6.4.2 Exemple

Avant de présenter les détails de la méthode, nous proposons un exemple (voir figure 6.1) qui permettra d'illustrer les différents types de réseaux utilisés pour la résolution du problème. Dans cet exemple, l'horizon d'optimisation est découpé en neuf périodes $t_1, \ldots, t_9$, deux pièces de travail $\{p_1, p_2\}$ sont planifiées et deux types d'activités $\{a_1, a_2\}$ sont possibles. La première pièce couvre les périodes t_1 à t_6 et quatre blocs peuvent être assignés à cette pièce :

- le bloc b_1 couvrant l'activité a_1 pour les périodes t_1 à t_3 ;
- le bloc b_2 couvrant l'activité a_1 pour les périodes t_1 à t_4 ;
- le bloc b_3 couvrant l'activité a_2 pour les périodes t_4 à t_6 ;
- le bloc b_4 couvrant l'activité a_2 pour les périodes t_5 à t_6.

La deuxième pièce couvre les périodes t_4 à t_9 et quatre blocs peuvent être assignés à cette pièce :

- le bloc b_5 couvrant l'activité a_1 pour les périodes t_4 à t_6 ;
- le bloc b_6 couvrant l'activité a_1 pour les périodes t_4 à t_7 ;

Algorithme 6.1 TabouSched : Algorithme de recherche tabou pour le problème d'attribution des activités.

1. Déterminer les périodes de sous-couverture et de sur-couverture inévitables ;
2. Appliquer les techniques de réduction ;
3. Générer une solution initiale $s_2 \in S_2$
4. Générer une solution initiale $s_1 \in S_1$ qui minimise z' pour s_2.
5. Établir $z^* = z'(s_1, s_2)$.

tant que la limite de temps n'est pas atteinte **effectuer**

6. Obtenir la meilleure solution non taboue $(x, y) \in S_1 \times S_2$ dans le voisinage de (s_1, s_2).
7. Obtenir la meilleure solution $(x', y') \in S_1 \times S_2$ dans le voisinage de (s_1, s_2).

si $z'(x', y') < z^*$ **alors**

8. $(s_1, s_2) = (x', y')$.

sinon

9. $(s_1, s_2) = (x, y)$.

fin si

10. Mettre à jour les graphes permettant d'explorer S_1 et S_2 ;
11. Mettre à jour la liste tabou L.

fin tant que

- le bloc b_7 couvrant l'activité a_2 pour les périodes t_7 à t_9 ;
- le bloc b_8 couvrant l'activité a_2 pour les périodes t_8 à t_9.

La demande pour l'activité a_1 est de 0 pour les périodes t_8 et t_9, de 1 pour les périodes t_1 à t_3 et t_5 à t_7 et de 2 pour la période t_4. La demande pour l'activité a_2 est de 0 pour les périodes t_1 à t_4 et pour la période t_7 et de 1 pour les autres périodes. Notons que l'ensemble des blocs permis a été déterminé a priori et n'est pas généré par les durées minimales et maximales $d_{min,a,t}$ et $d_{max,a,t}$ qui ne sont d'ailleurs pas définies.

La seule solution s^* qui permet de couvrir la demande est d'utiliser les blocs b_2, b_4, b_6 et b_8.

6.4.2.1 Initialisation

La phase d'initialisation a deux objectifs, soit de déterminer les périodes où les sous-couvertures et les sur-couvertures sont inévitables et trouver une solution initiale réalisable. Ces objectifs correspondent à la première et la troisième étape de l'algorithme 6.1.

6.4.2.2 Sous-couvertures et sur-couvertures inévitables

Nous cherchons ici simplement à déterminer si, pour chaque période t, la somme des demandes $\delta_t = \sum_{a \in A} \delta_{at}$ est égale au nombre de quarts q_t faisant de la couverture à cette période. Nous définissons le nombre de sous-couvertures inévitables pour la période t : $u_t = \max\{0, \delta_t - q_t\}$ et le nombre de sur-couvertures inévitables pour la période t : $o_t = \max\{0, q_t - \delta_t\}$. On voit que le nombre total des sous-couvertures moins le nombre total de sur-couvertures est égal à $\sum_{t \in T}(u_t - o_t)$, c'est à dire que si le nombre de sous-couvertures est $u_t + k$ alors le nombre de sur-couvertures est $o_t + k$. Pour vérifier cette affirmation, on définit, pour la période t, $\delta_{t,c}$ le nombre de demandes couvertes, $\delta_{t,u}$ le nombre de demandes non couvertes, $q_{t,c}$ le nombre de quarts couvrant une demande, $q_{t,o}$ le nombre de quarts faisant de la sur-couverture, et on suppose que le nombre de sur-couvertures est $o_t + k'$. Puisque le nombre d'activités couvertes égale le nombre de quarts couvrant des activités, on a $q_{t,c} = \delta_{t,c}$, posons $u_t + k = \delta_{t,u} = \delta_t - \delta_{t,c}$ et $o_t + k' = q_{t,o} = q_t - q_{t,c}$ alors $u_t + k - (o_t + k') = \delta_t - q_t + q_{t,c} - \delta_{t,c} = u_t - o_t$ ainsi $k = k'$. Dans notre algorithme nous traiterons de façon spéciale ces sous et sur-couvertures inévitables et elles ne seront pas considérées dans l'objectif. Ainsi le nombre de sous-couvertures potentiellement évitables est identique au nombre de sur-couvertures potentiellement évitables.

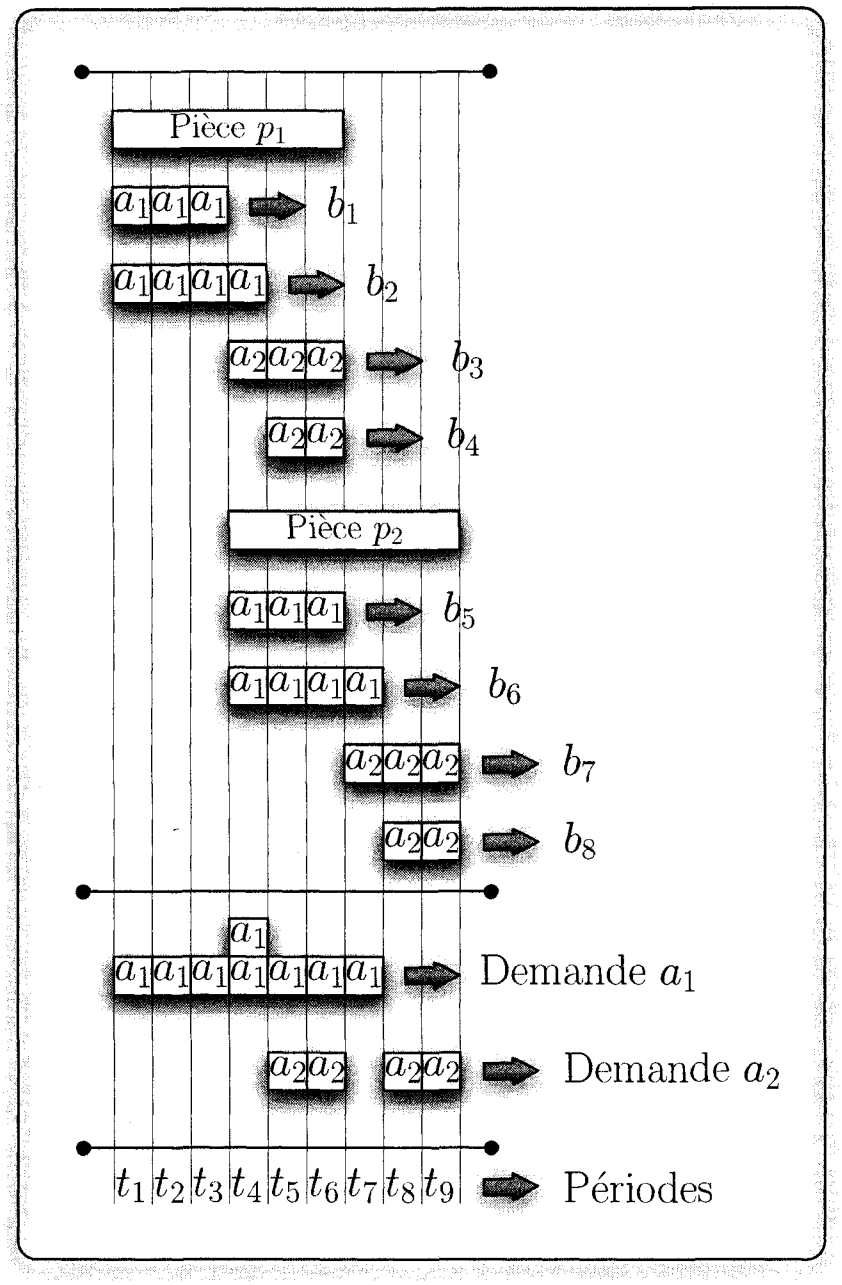
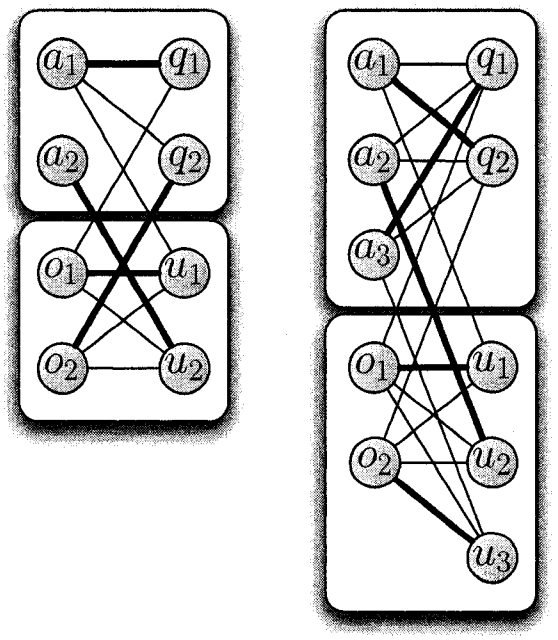


Figure 6.1 – Exemple d'un problème d'assignation d'activités à des quarts.

6.4.2.3 Solution initiale

La solution initiale s_2 est obtenue par affectation successive, c'est-à-dire les affectations sont effectuées successivement, une période après l'autre, en commencant par la première. À chaque période est associé un problème de couplage dans un graphe biparti $G_t(V_1, V_2, E)$, un couplage parfait de poids minimal détermine les attributions d'activités pour la période t. Les ensembles V_1 et V_2 sont divisés en deux parties, soit la partie de couverture et la partie de sous-couverture (voir figure 6.2). Dans la partie de couverture, on retrouve, dans V_1, un nœud pour chaque unité de demande à la période t (ensemble V_a) et, dans V_2, un nœud pour chaque quart disponible pour faire de la couverture à la période t (ensemble V_q). Dans la partie de sous-couverture, on retrouve, dans V_2, un nœud pour chaque unité de demande à la période t (ensemble V_u) et, dans V_1, un nœud pour chaque quart disponible pour de la couverture à la période t (ensemble V_o). Il existe une arête (i, j), $i \in V_a$ et $j \in V_q$ si l'affectation de l'activité correspondant à la demande du nœud i est possible pour le quart associé à j. Ces arêtes représentent l'affectation de l'activité associée à i au quart associé à j pour la période considérée. On ajoute aussi, pour chaque unité de demande, une arête entre le représentant de cette demande dans V_a et celui dans V_u. Ces arêtes permettent la sous-couverture de cette demande. De façon similaire, il existe une arête, pour chaque quart disponible pour de la couverture à cette période, entre le représentant de ce quart dans V_q et celui dans V_o. Ces arêtes permettent la sur-couverture. Il est à noter qu'il n'y a pas d'activité associée à cette sur-couverture. La nature de ces affectations est déterminée à la fin en essayant de minimiser la non réalisabilité. Finalement, le sous-graphe induit par les ensembles de nœuds V_o et V_u est un graphe biparti complet afin d'assurer qu'il existe un couplage parfait dans la partie de sous-couverture. Pour privilégier la couverture à la sous-couverture, un poids de un est mis sur les arêtes entre V_a et V_u. Le problème revient à trouver un couplage parfait de poids minimal.

La figure 6.2a illustre un exemple où, pour une période, il y a une demande pour l'activité a_1, une demande pour l'actvité a_2 et où les quarts q_1 et q_2 n'ont les compétences que pour la tâche a_1. La solution représentée par les arêtes en plus gros traits correspond à l'assignation (q_1, a_1) et $(q_2,$ à déterminer). Cet exemple illustre un autre type de sous-couverture/sur-couverture inévitable (dans des cas comme celui-là, chaque sous-couverture sera accompagnée d'une sur-couverture) dont l'on tiendra compte. La figure 6.2b est un autre exemple, où dans ce cas, il y a sous-couverture inévitable, la solution représentée est (q_1, a_3) et (q_2, a_1). Pour finir, notons, que l'on obtient ainsi, pour chaque période, une borne inférieure sur le nombre de sous-couvertures pour cette période.



a) b)

Figure 6.2 – Deux exemples de réseau pour une période lors de l'assignation initiale.

Pour trouver s_1 on résoudra un problème de plus court chemin pour chaque pièce de

travail. Pour la pièce p du quart q, ce problème est défini sur un graphe $G_p(V, E)$ (voir figure 6.3) qui contient, un nœud de départ, un nœud de fin et pour chaque période t de la pièce et pour chaque activité $a \in A_q$ les nœuds v^s_{at} et v^e_{at} qui représentent, respectivement, le début (s) et la fin (e) de l'activité a à la période t. Il y a un arête $e = (v^s_{at}, v^e_{at'})$ si il existe un bloc $b \in S_{pat} \cap E_{pat'}$. Cette arête correspond à assigner l'activité a à la pièce p pour les périodes t à t'. Le coût de cette arête est $(c_{uc} + c_{oc})$ multiplié par le nombre de périodes d'assignation dans la solution $s_2 \in S_2$ pour la pièce p qui ne sont pas a entre t et t', soit $c(e) = (c_{uc}+c_{oc})|\{t''|t \leq t'' \leq t' \wedge s_{2,aqt''} = 0\}|$ où $s_{2,aqt''}$ vaut 1 si l'assignation de la solution s_2 à la période t'' pour le quart q est a et vaut 0 sinon.

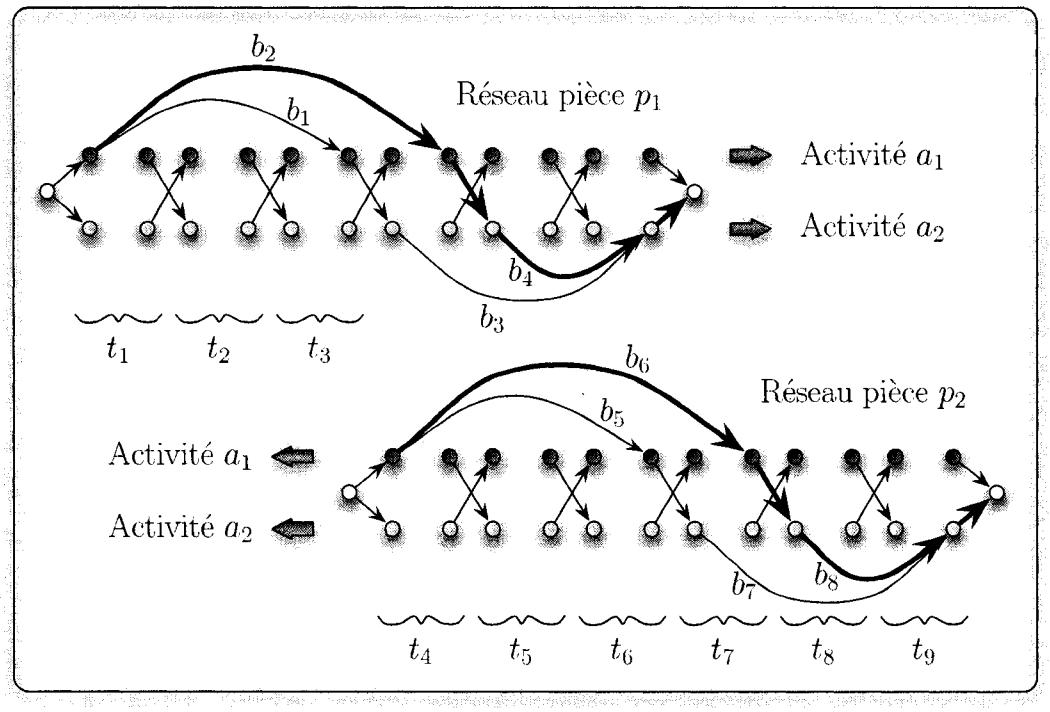


Figure 6.3 – Réseaux pour les quarts de l'exemple.

De plus, il y a une arête de coût c_t entre chaque paire de nœuds $(v^f_{at}, v^s_{a't+1}), a \neq a'$, qui correspond à faire une transition entre une activité de type a se terminant à la

période t et une activité de type a' commençant à la période $t+1$. Finalement il y a des arêtes de coût 0 entre le nœud de départ et les nœuds $v^s_{a,t_{deb,p}}$ lorsque $t_{deb,p}$ est la première période de la pièce et il y a des arêtes de coût 0 entre les nœuds $v^e_{a,t_{fin,p}}$ et le nœud de fin lorsque $t_{fin,p}$ est la dernière période de la pièce. En pratique les nœuds $v^s_{a,t_{deb,p}}$ et $v^e_{a,t_{fin,p}}$ sont fusionnés, respectivement, au nœud de départ et au nœud de fin. La solution s_1 est celle qui correspond, pour chaque graphe de pièce de travail, au plus court chemin dans ce graphe.

Par exemple supposons que l'on cherche à résoudre l'exemple présenté au début de cette section, la figure 6.3 représente les réseaux pour les deux pièces. La solution s^* est représentée par les arêtes en traits plus épais. Supposons aussi que la solution s_2 est la suivante : pour la pièce p_1 on a le couple (t_1, a_1) représentant l'assignation de l'activité a_1 à la période t_1, de la même façon on a (t_2, a_1), (t_3, a_1), (t_4, a_1), (t_5, a_2) et (t_6, a_1) et pour la pièce p_2 on a (t_4, a_1),(t_5, a_1), (t_6, a_2), (t_7, a_1), (t_8, a_2) et (t_9, a_2). Si $c_{uc} + c_{oc}$ vaut 1000, les coûts des arêtes sont $c(b_1) = 0$ pour l'arête associée au bloc b_1, $c(b_2) = 0$, $c(b_3) = 2000$, $c(b_4) = 1000$, $c(b_5) = 1000$, $c(b_6) = 1000$, $c(b_7) = 1000$, $c(b_8) = 0$, le coût est c_t pour les arêtes de transition et 0 pour les autres arêtes. Le plus court chemin a un coût de $2000 + 2c_t$ et correspond à la solution c^*.

6.4.3 Voisinage

Pour parler des voisinages, nous devons d'abord définir les graphes dans lesquels les solutions de S_1 et de S_2 seront exprimées. Pour S_1, les graphes sont identiques à ceux utilisés lors de la phase d'initialisation sauf que les coûts sont déterminés en utilisant la solution courante s_2 au lieu de la solution initiale. Pour S_2, on définit un graphe biparti pour chaque période. Les graphes de période sont très similaires aux graphes de périodes vus dans la phase d'initialisation. En fait, si il n'y a pas de sous ou sur-couvertures inévitables, ceux-ci correspondent au sous-graphe induit

par les ensembles de nœuds V_a et V_q (voir figure 6.4a). Dans le cas où il y a k sous-couvertures inévitables, on ajoute k nœuds de sous-couverture. Chacun de ces nœuds est relié à tous les nœuds de l'ensemble V_a (voir figure 6.4b où on a une demande pour a_1, deux demandes pour a_2 et seulement deux quarts disponibles). Dans le cas où il y a k sur-couvertures inévitables, on ajoute k nœuds de sur-couverture. Chacun de ces nœuds est relié à tous les nœuds de l'ensemble V_q (voir figure 6.4c où on a une demande pour a_1, pour a_2 et pour a_3 et trois quarts doivent être remplis). Finalement, si k sous-couvertures/sur-couvertures inévitables ont été détectées lors de la phase d'initialisation (on rappelle que, dans ces cas, chaque sous-couverture est associée à une sur-couverture) on ajoute, de la même façon que décrit précédemment, k nœuds de sous-couverture et k nœuds de sur-couverture. Chaque arête de ce graphe représente soit l'assignation d'une demande d'activité à un quart, l'assignation d'une demande d'activité à une sous-couverture ou l'assignation d'une sur-couverture à un quart.

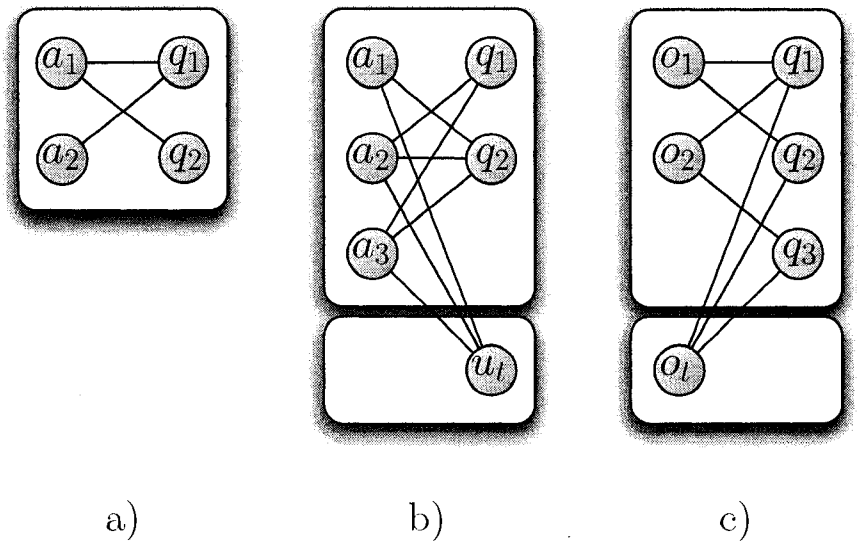


Figure 6.4 – Trois exemples de réseau pour une période.

Le voisnage de la solution courante (s_1, s_2) comportera un voisin pour chaque période t de l'horizon d'optimisation T. Afin d'exprimer les attributions effectuées dans une solution s, nous définissons la fonction d'affectation d'une pièce $f_{s,p}(t) : T_p \rightarrow A_q$ où T_p est l'ensemble des périodes où la pièce p est disponible pour de la couverture. Cette fonction détermine l'activité affectée à la pièce p du quart q à la période t. La solution (s'_1, s'_2) de ce voisinage pour la période t a les propriétés suivantes :

- Les attribution d'activités dans s'_2 et dans s_2 ne diffèrent que pour la période t. Ainsi, $f_{s'_2,p}(t') = f_{s_2,p}(t')$ pour tout $t' \neq t$ et pour toutes les pièces $p \in P$.
- s'_2 et s_2 ne sont pas identiques, i.e. il existe au moins une pièce p telle que $f_{s'_2,p}(t) \neq f_{s_2,p}(t)$.
- On assigne une pièce p appartenant à l'ensemble P_t des pièces disponibles pour faire de la couverture à la période t la même activité pour la période t dans s'_1 et dans s'_2. On a donc $p \in P_t \Rightarrow f_{s'_1,p}(t) = f_{s'_2,p}(t)$.
- Les attribution d'activités dans s'_1 et dans s_1 sont identiques pour les pièces qui ne sont pas disponibles pour de la couverture à la période t. Ainsi, $p \notin P_t \Rightarrow f_{s'_1,p}(t') = f_{s_1,p}(t'), \forall t' \in T_p$.
- La paire de solutions (s'_1, s'_2) respecte les contraintes précédentes et minimise $z'(s'_1, s'_2)$.

La solution (s'_1, s'_2) pour la période t est obtenue en résolvant d'abord un couplage de coût minimum dans le réseau G_t de la période t pour obtenir s'_2. Ensuite, pour chaque pièce de travail $p \in P_t$ nous créons le graphe G'_{pta} où $a = f_{s'_2,p}(t)$ en modifiant le graphe G_p afin de forcer l'attribution de l'activité $f_{s'_2,p}(t)$ à la pièce p à la période t. Pour ce faire, on enlève du graphe G_p toutes les arêtes qui n'assignent pas l'activité $f_{s'_2,p}(t)$ à la pièce p à la période t. Ensuite, si $f_{s_2,p}(t) \neq f_{s'_2,p}(t)$, on soustrait $c_{uc} + c_{oc}$ au coût des arêtes qui assignent l'activité $f_{s'_2,p}(t)$ à la pièce p à la période t afin d'ajuster le coût des arêtes à la solution s'_2. Les coûts des arêtes de G_t doivent être déterminés de sorte à ce que la valeur d'un couplage dans ce graphe corresponde

à la variation de $z'(s'_1, s'_2)$ par rapport à la valeur courante $z'(s_1, s_2)$; où s'_2 est la solution associée au couplage et s'_1 la solution qu'il induit dans S_1. Soit e une arête du graphe G_t représentant l'assignation de l'activité a à la pièce p pour la période t. Si cette assignation ne correspond pas à l'assignation correspondante dans la solution courante s_2 on définit le coût de e de la façon suivante :

- On calcule le coût $c_1(spp(e))$ du plus court chemin $spp(e)$ du graphe modifié G'_{pta} décrit précédemment.
- On définit le coût $c(e) = c_1(spp(e)) - c_{s_1}(p)$ où $c_{s_1}(p)$ est le coût du plus court chemin de la solution courante s_1 dans le graphe G_p.

Les arêtes qui correspondent à la solution courante s_2 ou à des sous-couvertures ou des sur-couvertures ont un coût de zéro. Ainsi, le coût du couplage correspond donc à $z'(s'_1, s'_2) - z'(s_1, s_2)$ car le coût $c(e) = c_1(spp(e)) - c_{s_1}(p)$ de l'arête e correspond à la variation du coût de l'attribution pour la pièce p dans la nouvelle solution (s'_1, s'_2) par rapport à son coût dans la solution (s_1, s_2). Les affectations qui correspondent aux pièces n'appartenant pas à P_t restent inchangées dans s'_1 et s'_2 et ne modifient pas la valeur de z'.

6.4.4 Liste tabou

À chaque itération, on choisit un couplage dans le graphe G_t d'une période t. Suite à cela, on ajoute à la liste tabou les arêtes e appartenant à ce couplage et correspondant à une nouvelle assignation par rapport à la solution pécédente. Les arêtes taboues sont retirées du graphe G_t tant que leur statut tabou n'est pas levé.

6.4.5 Réduction de la taille du problème

Dans tous les cas, on peut enlever les blocs qu'il est impossible d'utiliser dans une pièce de travail. Pour ce faire, on résout pour chaque $b' \in B$ appartenant à la pièce p du quart q un problème de plus court chemin. Les réseaux utilisés sont les mêmes que ceux développés pour trouver la solution initiale s_1 et pour lesquels un exemple est présenté à la figure 6.3. Cependant, pour la réduction, les coûts de toutes les arêtes sont nuls à l'exception de l'arête représentant le bloc b' qui a un coût de -1. Un plus court chemin de coût -1 dans ce graphe correspond à une assignation pour la pièce p utilisant le bloc b'. On résout ce problème de plus court chemin et on enlève le bloc b' si la valeur optimale n'égale pas -1. Dans l'exemple de la sous-section 6.4.2 pour lequel les réseaux sont illustrés à la figure 6.3, la réduction n'enlèverait aucun bloc. Cependant, si l'on ajoute à la pièce p_1 un bloc couvrant l'actvité a_1 pour les périodes t_2, t_3 et t_4, on peut se convaincre qu'il n'y aurait pas de chemin passant par l'arête qui serait ajoutée pour représenter ce nouveau bloc. Le modèle de programmation linéaire associé à ce problème de flot est le suivant :

$$\textbf{F1'} \begin{cases} \min \quad -x_{b'} & (6.2) \\ \text{sujet à :} & \\ \displaystyle\sum_{a \in A_q} \sum_{b \in S_{pa,t_{deb,p}}} x_b = 1 & (6.3) \\ \displaystyle\sum_{b \in E_{pa,t-1}} x_b - \sum_{a' \in A_q \setminus \{a\}} \tau_{aa',t} = 0, \ \forall a \in A_q, \forall t \in T'_p & (6.4) \\ \displaystyle\sum_{a' \in A_q \setminus \{a\}} \tau_{a'a,t} - \sum_{b \in S_{pa,t}} x_b = 0, \ \forall a \in A_q, \forall t \in T'_p & (6.5) \\ x_b \in \{0,1\}, \forall b \in B_p & (6.6) \\ \tau_{aa'} \in \{0,1\}, \forall a, a' \in A_q, a \neq a', \forall t \in T'_p. & (6.7) \end{cases}$$

où on a :

- T_p, l'ensemble des périodes appartenant à la pièce p ;
- $T'_p = T_p \backslash \{t_{deb,p}\}$;
- B_p, l'ensemble des blocs d'activité de la pièce p ;
- $t_{deb,p}$, la première période de la pièce p ;
- $\tau_{aa',t}$, vaut 1 si il y a transition d'une activité a se terminant à la période $t-1$ vers une activité a' commençant à la période t, et 0 sinon.

La contrainte (6.3) demandent qu'une activité commence au début du quart, alors que les équations de conservation de flot (6.4)-(6.5) imposent que, lorsqu'un bloc d'un type d'activité se termine avant la fin de la pièce, il doit être suivi par un bloc d'un autre type d'activité. Il est possible de se débarasser des $|A_q|(|A_q|-1)(|T_p|-1)$ variables τ en ajoutant $|A_q|(|T_p|-1)$ contraintes selon le modèle suivant :

$$\textbf{F1}\begin{cases} \min \quad -x'_b & (6.8) \\ \text{sujet à :} & \\ \displaystyle\sum_{a\in A_q}\sum_{b\in S_{pa,t_{deb,p}}} x_b = 1 & (6.9) \\ \displaystyle\sum_{a\in A_q}\sum_{b\in E_{pa,t-1}} x_b - \sum_{a\in A}\sum_{b\in S_{pat}} x_b = 0, \ \forall t \in T'_p & (6.10) \\ \displaystyle\sum_{b\in E_{pa,t-1}} x_b - \sum_{a'\in A_q\backslash\{a\}}\sum_{b\in S_{pa't}} x_b \leq 0, \ \forall a \in A_q, t \in T'_p & (6.11) \\ x_b \in \{0,1\}, \forall b \in B_p. & (6.12) \end{cases}$$

Ce modèle est obtenu en fusionnant ensemble les nœuds d'activités différentes correspondant à une même période, puis en éliminant le nœud d'arrivée aggloméré de la période t en le fusionnant au nœud de départ aggloméré de la période $t+1$. Cette transformation a pour effet d'éliminer du graphe le mécanisme empêchant que deux blocs d'une même activité se suivent immédiatement dans une même pièce. Ce mécanisme a été remplacé par les contraintes (6.11). La proposition suivante montre que

les relaxations linéaires de ces deux formulations sont, d'une certaine façon, équivalentes :

Proposition 12. *le vecteur* $x = \{x_b\}, \forall b \in B_p$ *est une solution de la relaxation linéaire de* ***F1*** *si et seulement si il existe un vecteur* $\tau = \{\tau_{aa',t}\}, \forall a, a' \in A_q, a \neq a', t \in T'_p$ *tel que* (x, τ) *est solution de la relaxation linéaire du modèle* ***F1'****.*

Démonstration. $\Leftarrow$ Soit (x, τ) une solution de **F1'**. Puisque le vecteur x respecte la contrainte (6.3) il respecte forcément la contrainte (6.9) qui est identique, de plus, dans les deux relaxations on demande $0 \leq x_b \leq 1, \forall b \in B_p$. Pour vérifier le respect des contraintes (6.10) et (6.11), nous définissons Γ^1_{at} (resp. Γ^2_{at}) le terme correspondant au membre de gauche de la contrainte (6.4) (resp. (6.5)) associée à l'activité a et à la période t et Υ^1_t le terme correspondant au membre de gauche de la contrainte (6.10) associée à la période t, on a :

$$\Upsilon^1_t = \sum_{a \in A_q} (\Gamma^1_{at} + \Gamma^2_{at}) \tag{6.13}$$

$$\Gamma^1_{at} + \sum_{a' \in A_q \backslash \{a\}} \Gamma^2_{a't} = \sum_{b \in E_{pa,t-1}} x_b - \sum_{a' \in A_q \backslash \{a\}} \sum_{b \in S_{pa't}} x_b + \sum_{a' \in A_q \backslash \{a\}} \sum_{a'' \in A_q \backslash \{a,a'\}} \tau_{a'a''} \tag{6.14}$$

Puisque les termes de droite des contraintes (6.4) et (6.5) sont nuls, l'équation (6.13) implique que $\Upsilon^1_t = 0$ et donc que le vecteur x respecte les contraintes (6.10) pour tout $t \in T_p \backslash \{t_{deb,p}\}$. On conclut aussi, par l'équation (6.14), que $\sum_{b \in E_{pa,t-1}} x_b - \sum_{a' \in A_q \backslash \{a\}} \sum_{b \in S_{pa't}} x_b + \sum_{a' \in A_q \backslash \{a\}} \sum_{a'' \in A_q \backslash \{a,a'\}} \tau_{a'a''} = 0$. Puisque $\tau \geq 0$, on a $\sum_{b \in E_{pa,t-1}} x_b - \sum_{a' \in A_q \backslash \{a\}} \sum_{b \in S_{pa't}} x_b \leq 0$ et, par conséquent, x respecte les contraintes (6.11) pour tout

$a \in A$ et $t \in T'_p$. Le vecteur x est une solution réalisable de la relaxation linéaire de **F1**.

$\Rightarrow$ Il reste à montrer que pour tout vecteur solution x de la relaxation linéaire de **F1** il existe un vecteur τ tel que (x, τ) est une solution réalisable de la relxation linéaire de **F1'**. Comme précédemment, le vecteur x respecte la contrainte (6.3) car il respecte la contrainte (6.9) qui est identique et dans les deux cas $0 \leq x_b \leq 1, \forall b \in B_p$. Soit t une période quelconque de T'_p : on définit $f_{a,t-1} = \sum_{b \in E_{pa,t-1}} x_b$ et $f_{a,t} = \sum_{b \in S_{pa,t}} x_b$; considérons maintenant le problème suivant

$$z = \min \sum_{a \in A_q} ((f_{a,t-1} - \varphi_{a,t-1}) + (f_{a,t} - \varphi_{a,t})) \tag{6.15}$$

sujet à :

$$f_{a,t-1} - \varphi_{a,t-1} \geq 0, \ \forall a \in A_q \tag{6.16}$$

$$\varphi_{a,t} - f_{a,t} \leq 0, \ \forall a \in A_q, \tag{6.17}$$

$$0 \leq \tau_{a'a,t} \leq 1, \forall a', a \in A_q, a \neq a' \tag{6.18}$$

où $\varphi_{a,t-1} = \sum_{a' \in A_q \setminus \{a\}} \tau_{aa',t}$ et $\varphi_{a,t} = \sum_{a' \in A_q \setminus \{a\}} \tau_{a'a,t}$. Les contraintes (6.16) et (6.17) sont des relaxations des contraintes de conservation de flot (6.4) et (6.5) où l'on permet que le flot $f_{a,t-1}$ (qui est fixe) entrant dans le nœud v^f_{at-1} soit plus grand que le flot sortant $\varphi_{a,t-1}$ de ce nœud. De façon similaire on permet que le flot $f_{a,t}$ (qui est fixe) sortant du nœud $v^e_{a,t}$ soit plus grand que le flot entrant $\varphi_{a,t}$ de ce nœud. La fonction objectif à pour but de minimiser l'écart entre les flots entrants et sortants. Dans le meilleur des cas, il y a conservation de flot et la valeur de l'objectif est 0, notons qu'inversement, une valeur de l'objectif de 0 implique une conservation de flot. Le vecteur trivial $\tau = 0$ est toujours solution de ce problème car les flots entrants $f_{a,t-1}$

et sortants $f_{a,t}$ sont positifs ou nuls. Donc, (x, τ) est une solution de **F1'** si τ est une solution optimale du problème (6.15)-(6.18) et que la valeur optimale est $z = 0$. Les trois observations suivantes seront utiles pour le reste de la démonstration :

$$\sum_{a \in A_q} \varphi_{a,t-1} = \sum_{a \in A_q} \varphi_{a,t} = \sum_{a,a' | a \neq a'} \tau_{aa',t} \tag{6.19}$$

$$\sum_{a \in A_q} f_{a,t-1} = \sum_{a \in A_q} f_{a,t} \quad \text{par la contrainte (6.10)} \tag{6.20}$$

$$\sum_{a \in A_q} (f_{a,t-1} - \varphi_{a,t-1}) = \sum_{a \in A_q} (f_{a,t} - \varphi_{a,t}) \quad \text{par (6.19) et (6.20)} \tag{6.21}$$

Montrons par l'absurde qu'il existe toujours un vecteur τ pour lequel $z = 0$. Soit τ une solution optimale du problème, si $z > 0$ alors $\sum_{a \in A_q} \varphi_{a,t-1} < \sum_{a \in A_q} f_{a,t-1}$, et il existe une activité a telle que $\varphi_{a,t-1} < f_{a,t-1}$. Si il existe deux activités a' et a'' telles que $a' \neq a''$, $\varphi_{a',t-1} < f_{a',t-1}$ et $\varphi_{a'',t} < f_{a'',t}$ alors on peut augmenter le flot $\tau_{a'a''}$ de $\delta = \min\{f_{a',t-1} - \varphi_{a',t-1}, f_{a'',t} - \varphi_{a'',t}\} > 0$ tout en laissant les autres flots inchangés, ce qui fait diminuer la valeur de l'objectif de $\delta > 0$, par conséquent τ n'est pas optimal, une contradiction. Si ce n'est pas le cas alors pour toute activité $a' \neq a$ on a $\varphi_{a',t} = f_{a',t}$ donc 6.21 implique $\varphi_{a,t} < f_{a,t}$. L'inéquation (6.11) établit que $\varphi_{a,t-1} < f_{a,t-1} \leq \sum_{a' \in A_q \setminus \{a\}} f_{a',t}$ et puisque si $a' \neq a$ on a $\varphi_{a',t} = f_{a',t}$ l'inégalité devient $\varphi_{a,t-1} < \sum_{a' \in A_q \setminus \{a\}} \varphi_{a',t}$ et par conséquent il existe une actvité a' telle que $\varphi_{a',t} > \tau_{aa'}$ ce qui implique l'existence d'une activité a'' telle que $\tau_{a''a'} > 0$. Ainsi les modifications $\tau_{aa'} = \tau_{aa'} + \delta, \tau_{a''a'} = \tau_{a''a'} - \delta, \tau_{a''a} = \tau_{a''a} + \delta$ où $\delta = \min\{f_{a,t-1} - \varphi_{a,t-1}, f_{a,t} - \varphi_{a,t}, \tau_{a''a'}\} > 0$ respectent les contraintes et diminuent l'objectif de $\delta > 0$, encore là τ n'est pas optimal et on a une contradiction. Alors une solution optimale au problème (6.15)-(6.18) produit la solution (x, τ) recherchée, ce qui complète la démonstration d'équivalence entre les deux formulations. □

Dans le cas où l'on sait qu'il existe une solution sans sous-couverture (ou que l'on suppose qu'il en existe une) et que l'on minimise d'abord la sous-couverture avant les transitions ($c_{uc} \ggg c_t$), il est possible d'enlever un certain nombre de blocs en vérifiant pour chacun si il est possible de répondre exactement à la demande en utilisant ce bloc. Pour ce faire, on résout pour chaque $b \in B$ (en supposant $b \in B_{pa}$) un problème de flot à coût minimum. On associe pour l'activité $a \in A$ un multi-graphe orienté $G_a(V, E)$ (voir figure 6.5). Dans ce graphe, on définit un nœud de fin i_{fin} et, pour chaque période $t \in T$, un nœud i_t. Soit un bloc b' commençant à la période t_1 et se terminant à la période t_2, si $t_2 \neq t_{fin}$, on aura une arête $e = (i_{t_1}, i_{t_2+1})$ ou, si $t_2 = t_{fin}$, une arête $e = (i_{t_1}, i_{fin})$. Le coût de cette arête est 0, les capacités minimale et maximale de flot sur cette arête sont respectivement de 0 et de 1. Pour chaque paire de nœuds i_t, i_{t+1} et pour la paire $i_{t_{fin}}, i_{fin}$, on a une arête (i_t, i_{t+1}) (resp. $(i_{t_{fin}}, i_{fin})$) de coût c_{uc} correspondant à une sous-couverture à la période t (resp. t_{fin}) et une arête (i_{t+1}, i_t) (resp. $(i_{fin}, i_{t_{fin}})$) de coût c_{oc} correspondant à une sur-couverture à la période t (resp. t_{fin}). La capacité minimale de flot sur ces arêtes est de 0. Le nœud i_t est une source (ou un puits) fournissant (ou acceptant) $\delta_{at} - \delta_{at-1}$ (δ_{at} si $t = t_{deb}$) unités de flot. Pour savoir si l'on peut éliminer le bloc b, on modifie le problème de flot associé au graphe G_a en attribuant un coût $c(e) = -1$ à l'arête e correspondant au bloc b, on résout le problème de flot à coût minimum de ce graphe modifié et on enlève le bloc b si la solution n'égale pas -1. Notons qu'on suppose $c_{uc} > 0$ et $c_{oc} > 0$, ainsi il est impossible d'avoir une solution de coût -1 si l'on utilise une arête correspondant à une sous-couverture ou à une sur-couverture.

Le modèle de programmation linéaire associé à ce problème de flot est le suivant :

$$\mathbf{F2}\begin{cases} \min & \sum_{a\in A,t\in T} c_{uc}u_{at} + \sum_{a\in A,t\in T} c_{oc}o_{at} - x_b & (6.22) \\ \text{sujet à :} & \\ & \sum_{b'\in S_{a,t_{deb}}} x_{b'} + u_{a,t_{deb}} - o_{a,t_{deb}} = \delta_{a,t_{deb}} & (6.23) \\ & \sum_{b'\in E_{at-1}} x_{b'} + u_{a,t-1} + o_{at} - \sum_{b'\in S_{at}} x_{b'} - o_{a,t-1} - u_{at} \\ & \qquad = \delta_{a,t-1} - \delta_{at}, \forall t \in T\backslash\{t_{deb}\} & (6.24) \\ & \sum_{b'\in E_{a,t_{fin}}} x_{b'} + u_{a,t_{fin}} - o_{a,t_{fin}} = \delta_{a,t_{fin}} & (6.25) \\ & x \in \{0,1\}, u, o \text{ entier} & (6.26) \end{cases}$$

où on a :

- t_{deb}, la première période de l'horizon d'optimisation ;
- t_{fin}, la dernière période de l'horizon d'optimisation ;
- S_{at} l'ensemble des blocs couvrant l'activité a et commençant à la période t ;
- E_{at} l'ensemble des blocs couvrant l'activité a et terminant à la période t ;

Si l'on considère l'exemple de la sous-section 6.4.2, les réseaux utilisés pour éliminer des blocs qui ne peuvent répondre exactement à la demande, lorsqu'on sait qu'il n'y a pas de sous-couvertures, sont illustrés à la figure 6.5. La solution s^* est représentée par les arêtes en traits plus épais. Pour le réseau de l'activité a_1, on pourrait enlever le bloc b_1 car les solutions réalisables ayant un flot de 1 sur cette arête doivent toutes avoir un flot de 1 sur l'arête correspondant à une sous-couverture pour la période t_4. On pourrait aussi grâce à cette procédure enlever les blocs b_3, b_5 et b_7. Il ne resterait alors que les blocs utilisés dans l'unique solution sans sous-couverture.

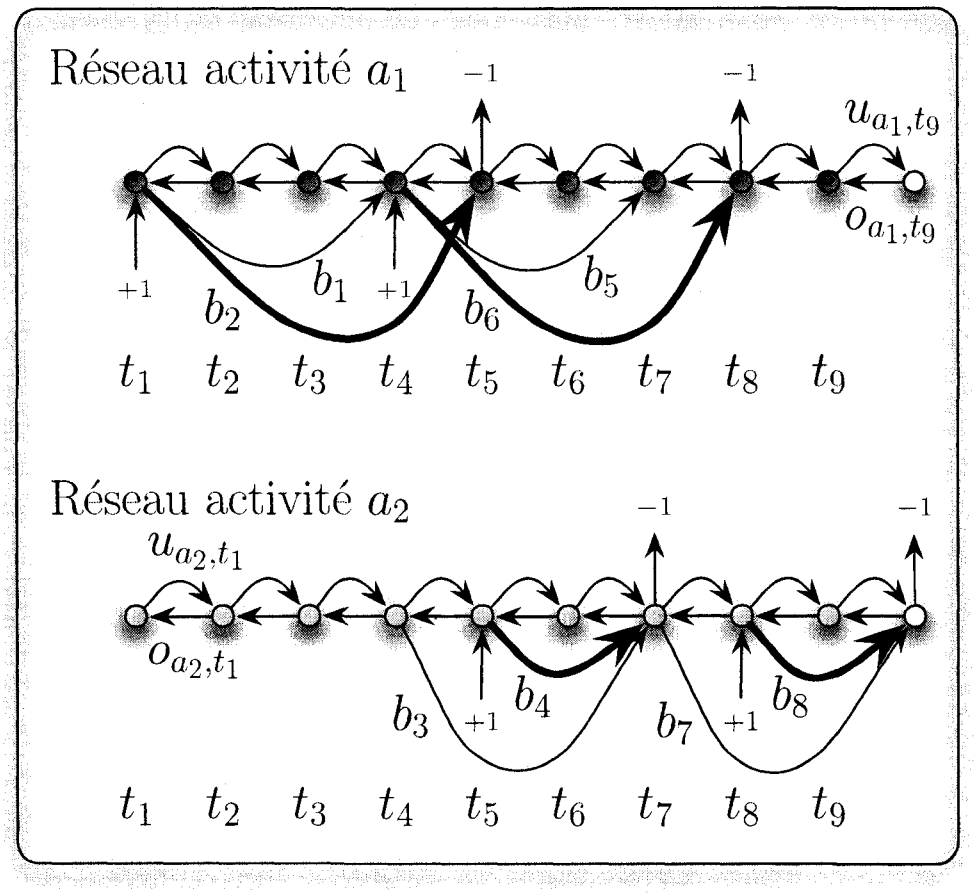


Figure 6.5 – Réseaux de demande de l'exemple.

6.4.6 Modèles de programmation en nombres entiers

La méthode proposée consiste à résoudre un problème de programmation linéaire en nombres entiers. Dans ce modèle, des variables représentent les blocs d'activités consécutives valides. Les autres variables permettent la sous-couverture et la sur-couverture d'activité. Afin de diminuer la taille du problème, nous résolvons dans un premier temps, une série de sous-problèmes pour éliminer des variables représentant des blocs d'activités. Ces variables correspondent aux arêtes des réseaux présentés

à la section précédente. C'est pourquoi on peut réutiliser les techniques d'élimination présentées dans cette section. Ensuite un logiciel de programmation en nombres entiers résout le modèle réduit. Plusieurs modèles sont présentés, et les expérimentations numériques menées auront pour but de comparer leur mérites respectifs. Pour commencer, nous proposons le modèle **M1** :

$$\mathbf{M1}\begin{cases} \min & \displaystyle\sum_{a\in A,t\in T} c_{uc}u_{at} + \sum_{a\in A,t\in T} c_{oc}o_{at} + \sum_{b\in B} c_t x_b - |P|\cdot c_t & (6.27)\\ \text{sujet à :} & & \\ & \displaystyle\sum_{p\in P}(\sum_{b\in B_{pa}} \alpha_{bat}x_b) + u_{at} - o_{at} = \delta_{at},\ \forall a\in A, \forall t\in T & (6.28)\\ & \displaystyle\sum_{a\in A}(\sum_{b\in B_{pa}} \alpha_{bat}x_b) = 1, \forall p\in P,\ \forall t\in T_p & (6.29)\\ & \displaystyle\sum_{b\in E_{pa,t-1}} x_b + \sum_{b\in S_{pat}} x_b \leq 1,\ \forall p\in P, a\in A, t\in T'_p & (6.30)\\ & x\in\{0,1\}, \forall b\in B_p & (6.31)\\ & u_{at}, o_{at} \text{ entier}, \forall a\in A, \forall t\in T. & (6.32) \end{cases}$$

Où $\alpha_{bat} = 1$ si le bloc b couvre l'activité a à la période $t \in T$, 0 sinon et les autres notations ont la même signification que précédemment. Les contraintes (6.28) assurent que soit la demande est exactement respectée, soit les pénalités de sous-couverture ou de sur-couverture sont appliquées. Les contraintes (6.29) assurent que, pour chaque pièce de travail, toutes les période de la pièce sont couvertes. Finalement, les contraintes (6.30) empêchent qu'un bloc couvrant une activité quelconque soit immédiatement suivi par un bloc couvrant cette même activité dans un même quart. Ces contraintes ne sont pas équivalentes aux contraintes (6.11) que nous avons utilisées à la section précédente dans le même but. Ces nouvelles contraintes quoique moins serrées introduisent moins d'éléments non-nuls dans la matrice de contraintes. Le prochain modèle **M2** consiste simplement à remplacer les contraintes (6.30) par les contraintes (6.11) plus serrées.

$$\mathbf{M2}\begin{cases}
\min & \sum_{a\in A,t\in T} c_{uc}u_{at} + \sum_{a\in A,t\in T} c_{oc}o_{at} + \sum_{b\in B} c_t x_b - |P|\cdot c_t & (6.33)\\
\text{sujet à :} & & \\
& \sum_{p\in P}(\sum_{b\in B_{pa}} \alpha_{bat}x_b) + u_{at} - o_{at} = \delta_{at}, \forall a \in A, \forall t \in T & (6.34)\\
& \sum_{a\in A}(\sum_{b\in B_{pa}} \alpha_{bat}x_b) = 1, \forall p \in P, \forall t \in T_p & (6.35)\\
& \sum_{b\in E_{pa,t-1}} x_b - \sum_{a'\in A, a'\neq a} \sum_{b\in S_{pa't}} x_b \leq 0, \forall a \in A, t \in T'_p & (6.36)\\
& x \in \{0,1\}, \forall b \in B_p & (6.37)\\
& u_{at}, o_{at} \text{ entier}, \forall a \in A, \forall t \in T. & (6.38)
\end{cases}$$

On peut observer que ce problème consiste à choisir des blocs d'activité qui vont à la fois former des quarts de travail valides tout en imposant les pénalités de sous-couverture et de sur-couverture, c'est-à-dire un ensemble de blocs qui est une solution qui respecte à la fois les contraintes des problèmes de flot **F1** pour chaque quart et les contraintes des problèmes de flot **F2** pour chaque activité. Suivant cette constatation, les deux prochains modèles sont obtenus en remplaçant les contraintes (6.28) et (6.29) par les contraintes de flot correspondantes, soit **M3** :

$$
\mathbf{M3}\begin{cases}
\min & \displaystyle\sum_{a\in A,t\in T} c_{uc}u_{at} + \sum_{a\in A,t\in T} c_{oc}o_{at} + \sum_{b\in B} c_t x_b - |P|\cdot c_t & (6.39)\\
\text{sujet à :} & \\
& \displaystyle\sum_{a\in A_q}\sum_{b\in S_{pa,t_{deb,p}}} x_b = 1,\ \forall p\in P & (6.40)\\
& \displaystyle\sum_{a\in A_q}\sum_{b\in E_{pa,t-1}} x_b - \sum_{a\in A}\sum_{b\in S_{pat}} x_b = 0,\ \forall p\in P, \forall t\in T'_p & (6.41)\\
& \displaystyle\sum_{b'\in S_{a,t_{deb}}} x_{b'} + u_{a,t_{deb}} - o_{a,t_{deb}} = \delta_{a,t_{deb}},\ \forall a\in A & (6.42)\\
& \displaystyle\sum_{b'\in E_{at-1}} x_{b'} + u_{a,t-1} + o_{at} - \sum_{b'\in S_{at}} x_{b'} - o_{a,t-1} - u_{at} & \\
& \qquad = \delta_{a,t-1} - \delta_{at},\ \forall a\in A, \forall t\in T' & (6.43)\\
& \displaystyle\sum_{b'\in E_{a,t_{fin}}} x_{b'} + u_{a,t_{fin}} - o_{a,t_{fin}} = \delta_{a,t_{fin}},\ \forall a\in A & (6.44)\\
& \displaystyle\sum_{b\in E_{pa,t-1}} x_b + \sum_{b\in S_{pat}} x_b \le 1,\ \forall p\in P, a\in A, t\in T'_p & (6.45)\\
& x\in\{0,1\}, \forall b\in B_p & (6.46)\\
& u_{at}, o_{at} \text{ entier}, \forall a\in A, \forall t\in T. & (6.47)
\end{cases}
$$

où $T' = T\backslash\{t_{deb}\}$. Puis, en remplaçant les contraintes (6.45) par les contraintes (6.11), on obtient **M4** :

$$
\mathbf{M4}\begin{cases}
\min & \displaystyle\sum_{a\in A,t\in T} c_{uc}u_{at} + \sum_{a\in A,t\in T} c_{oc}o_{at} + \sum_{b\in B} c_t x_b - |P|\cdot c_t & (6.48)\\
\text{sujet à :} & \\
& \displaystyle\sum_{a\in A_q}\sum_{b\in S_{pa,t_{deb,p}}} x_b = 1,\ \forall p\in P & (6.49)\\
& \displaystyle\sum_{a\in A_q}\sum_{b\in E_{pa,t-1}} x_b - \sum_{a\in A}\sum_{b\in S_{pat}} x_b = 0,\ \forall p\in P, \forall t\in T'_p & (6.50)\\
& \displaystyle\sum_{b'\in S_{a,t_{deb}}} x_{b'} + u_{a,t_{deb}} - o_{a,t_{deb}} = \delta_{a,t_{deb}},\ \forall a\in A & (6.51)\\
& \displaystyle\sum_{b'\in E_{at-1}} x_{b'} + u_{a,t-1} + o_{at} - \sum_{b'\in S_{at}} x_{b'} - o_{a,t-1} - u_{at} & \\
& \qquad = \delta_{a,t-1} - \delta_{at},\ \forall a\in A, \forall t\in T' & (6.52)\\
& \displaystyle\sum_{b'\in E_{a,t_{fin}}} x_{b'} + u_{a,t_{fin}} - o_{a,t_{fin}} = \delta_{a,t_{fin}},\ \forall a\in A & (6.53)\\
& \displaystyle\sum_{b\in_{pa,t-1}} x_b - \sum_{a'\in A_q\setminus\{a\}}\sum_{b\in S_{pa't}} x_b \le 0, & \\
& \qquad \forall p\in P, \forall a\in A_q, t\in T'_p & (6.54)\\
& x\in\{0,1\}, \forall b\in B_p & (6.55)\\
& u_{at}, o_{at} \text{ entier}, \forall a\in A, \forall t\in T. & (6.56)
\end{cases}
$$

Dans notre approche, la méthode d'élimination **F1** visant à éliminer les blocs qui ne peuvent faire partie d'un quart admissible est appliquée systématiquement, alors que la méthode d'élimination **F2** qui retire les blocs qui ne peuvent faire partie d'une solution sans sur-couverture et sans sous-couverture est optionnelle.

6.5 Expérimentations numériques

Afin d'évaluer les qualités des différents modèles que nous avons mis de l'avant, nous les appliquerons sur un certains nombre d'instances. Nous décrivons d'abord les instances utilisées, puis les résultats obtenus avec la méthode heuristique et ensuite les résultats obtenus avec l'approche de programmation linéaire en nombres entiers (PLNE) utilisant les modèles **M1**,**M2**,**M3** et **M4**. Nous terminons par une dicussion des avantages respectifs des méthodes étudiées.

6.5.1 Banc de test

Les instances utilisées pour l'expérimentation sont divisées en trois catégories selon la durée de l'horizon d'optimisation soit des problèmes d'une journée (96 périodes de 15 minutes), des problèmes de deux jours (192 périodes de 15 minutes) et des problèmes d'une semaine (672 périodes de 15 minutes). Chaque instance a été générée de façon aléatoire, c'est-à-dire que les heures de débuts des quarts, la longueur des pièces, les durées minimales et maximales des activités varient aléatoirement autour d'une valeur cible. Le tableau 6.1 présente les caractéristiques des instances. On y trouve le nom de l'instance, le nombre de périodes $|P|$, le nombre d'activités $|A|$ ainsi que le nombre moyen de qualifications par employé ($\overline{Act/Emp}$), le nombre de pièces $|P|$ ainsi que la durée moyenne des pièces ($\overline{duree}$), le nombre de blocs d'activité possibles $|B|$ après la réduction **F1** (après **F1**) ainsi qu'après la réduction **F2** (après **F2**) si elle a été appliquée. Ensuite suivent, exprimées en nombre de périodes, la plus petite durée minimale pour une activité $\min d_{min}$, la plus grande durée maximale pour une activité $\max d_{max}$, la moyenne des durées minimales $\overline{d_{min}}$ et la moyenne des durées maximales $\overline{d_{max}}$. Finalement on a la valeur de la relaxation linéaire RL pour les modèles **M2** et **M4** ainsi que pour les modèles **M1** et **M3** (**M1**,**M3**) et finalement

Tableau 6.1 – Description des instances.

Instance	|T|	|A| (Act/Emp)	|P| (durée)	|B| après F1 (après F2)	min d_{min}	max d_{max}	$\overline{d_{min}}$	$\overline{d_{max}}$	RL M2,M4 (M1,M3)	Écart int M2 M4 (M1,M3)
1JourA	96	10 (4.50)	100 (17.70)	13161 (4549)	4	16	5.80	16.00	2077.50	0.36%
1JourB	96	10 (4.98)	100 (17.88)	20232 (8057)	4	16	5.00	16.00	2095.92	1.60%
1JourC	96	10 (4.98)	100 (17.88)	18517	4	16	5.10	16.00	7004.21	0.44%
1JourD	96	12 (7.02)	100 (19.44)	48789	3	19	5.25	17.17	61706.29 (61699.23)	1.75% (1.76)%
1JourE	96	12 (7.84)	100 (19.38)	54016	3	19	5.17	16.58	44561.49	0.04%
1JourF	96	10 (4.74)	100 (17.86)	18193 (7417)	4	16	5.20	16.00		
1JourG	96	10 (4.34)	100 (17.85)	13046 (4580)	4	16	6.20	16.00		
1JourH	96	10 (4.54)	100 (19.38)	13466 (5185)	4	16	5.90	16.00		
2JoursA	192	15 (7.44)	198 (17.94)	34637	5	16	6.20	16.00	24723.74	3.27%
2JoursB	192	15 (7.44)	198 (17.94)	62879 (14502)	4	16	5.20	16.00	4074.95 (4074.98)	0.85% (0.85)%
2JoursC	192	15 (7.40)	198 (17.94)	33328	5	16	6.26	16.00	52124.59	2.63%
2JoursD	192	12 (7.64)	180 (19.60)	78226	3	18	6.08	15.33	170093.23 (170085.27)	0.04% (0.04)%
2JoursE	192	12 (7.32)	180 (19.51)	69533	3	17	6.00	17.58	143314.26 (143304.51)	0.29% (0.30)%
SemaineA	672	10 (9.46)	580 (19.43)	200707 (40697)	5	16	5.90	16.00	11903.14	0.31%
SemaineB	672	10 (9.54)	580 (19.21)	187743 (40176)	5	16	6.00	16.00	11443.38	0.41%
SemaineC	672	10 (9.46)	580 (19.43)	197143	5	16	6.00	16.00	33293.45	0.11%
SemaineD	672	12 (7.38)	680 (19.47)	289841	3	14	6.42	16.83	671750.11 (671450.38)	— (—)
SemaineE	672	12 (7.34)	680 (19.45)	264389	3	20	6.42	15.92	704829.97 (703601.99)	— (—)

l'écart d'intégrité $\frac{100(OPT-RL)}{OPT}$ pour **M2** et **M4** ainsi que pour **M1** et **M3** (**M1**,**M3**) où OPT est la valeur optimale. Pour alléger le tableau, les valeurs liées à la relaxation linéaire pour **M1** et **M3** ne sont pas présentées quand celles-ci sont identiques à la relaxation linéaire pour **M2** et **M4**.

L'approche heuristique basée sur la méthode tabou a été utilisée sur les instances d'une journée ayant aucune sous/sur-couverture, en utilisant les deux méthodes de réduction. Les instances 1JourF, 1JourG et 1JourH ont été ajoutées pour augmenter le nombre d'instances d'une journée sans sous/sur-couverture et ne sont résolues qu'avec l'approche heuristique basée sur la méthode tabou. Tous les tests ont été exécutés sur une machine Intel(R) Core(TM)2 cpu 6700/2.66GHz. Les résultats sont présentés au tableau 6.2. On y retrouve le nom de l'instance, le nombre d'itérations

Tableau 6.2 – Résultats pour l'approche heuristique basée sur la méthode tabou.

Modèle	Nb d'itérations	Nb de sous-couvertures dans la meilleure solution trouvée
1JourA	4805200	10
1JourB	4234700	13
1JourF	6732600	12
1JourG	4831000	9
1JourH	4518000	13

après 2 heures de calcul et le nombre de paires de sous/sur-couvertures de la meilleure solution trouvée. Les résultats montrent que cette approche arrive qu'une seule fois à descendre en dessous de 10 sous-couvertures. Il reste donc beaucoup de place à l'amélioration. De toutes les tentatives pour améliorer notre méthode celle qui a donné les meilleurs résultats est l'utilisation des méthodes de réduction basées sur la résolution de sous-problèmes simples. C'est cette constatation qui nous a incités à investiguer plus en profondeur les modèles de PLNE que nous avons présentés à la section pécédente.

Le tableau 6.3 présente les résultats des quatre modèles **M1** à **M4** sur 15 instances (5 pour chaque type de durée). Les modèles de PLNE ont été résolus en utilisant le logiciel Xpress-MP 2007b de Dash Optimization avec les paramètres par défaut pour la méthode de séparation et évaluation progressive avec les coupes par défaut sur le nœud racine. Le logiciel a effectué un pré-traitement pour réduire la taille de la matrice. Tous les tests ont été exécutés sur une machine Intel(R) Core(TM)2 cpu 6700/2.66GHz. Le temps maximal de calcul est de deux heures. Rappelons que la méthode de réduction **F1** a été appliquée sur chacune des instances, de plus, lorsque l'instance possédait une solution sans sur/sous-couverture, nous avons effectué un test supplémentaire en appliquant la réduction **F2** en conjonction avec le modèle **M3** (**M3+F2** dans le tableau) afin d'avoir un aperçu de l'impact de cette méthode. Le

tableau présente le nom de l'instance, le modèle utilisé, la valeur de la borne inférieure après l'application des coupes au nœud racine, la meilleure solution entière obtenue ainsi que la borne inférieure (BInf) si la solution n'est pas optimale, le nombre de nœuds de branchement, le nombre de rangées dans la matrice ainsi que le nombre de rangées après pré-traitement (pré-traitement), le nombre de colonnes dans la matrice ainsi que le nombre de colonnes après pré-traitement (pré-traitement), le nombre d'éléments non-nuls dans la matrice ainsi que le nombre d'éléments non-nuls après pré-traitement (pré-traitement) et finalement, le temps total, en secondes, ainsi que le temps, en secondes, pris pour résoudre la relaxation linéaire du nœud racine (RL). Notons que, pour l'instance 2JoursD, même si la valeur optimale n'a pas été trouvée à l'intérieur de la limite de deux heures, nous avons pu obtenir cette valeur optimale qui est de 170160 lors de tests plus longs.

Il est difficile d'établir à partir de ces résultats quels paramètres font qu'une instance est plus difficile qu'une autre. La durée de l'horizon, comme on pouvait s'en douter, a un effet certain. Cependant la différence la plus marquée est entre les trois premières instances et les quatrième et cinquième instances de chaque groupe de durée. Ces différences peuvent s'expliquer par le procédé qui a mené à leur création. Dans le cas des trois premières instances, des quarts de travail valides sont générés aléatoirement et la demande est construite de façon à correspondre exactement à l'offre des quarts. Dans certains cas des sous/sur-couvertures sont générées en perturbant légérement les durées minimales et maximales des activités. Le résultat est une courbe de demande qui présente peu de variations rapides proche de ce que l'on retrouve en pratique. Dans le cas des quatrième et cinquième instances les quarts sont remplis aléatoirement d'activité permises pour celui-ci, mais sans imposer que les durées minimales et maximales soient respectées. Un incitatif est toutefois appliqué pour favoriser les chances que deux périodes consécutives se voient attribuer la même activité. Dans ce cas, la courbe de demande est plus anarchique et plus éloignée des courbes de

Tableau 6.3 – Résultats pour les modèles de PLNE.

Instances	Modèle	Valeur après coupes	Valeur entière (BInf)	Nb de nœuds	Rangées (Pré-traitement)	Colonnes (Pré-traitement)	Non-nuls (Pré-traitement)	Temps total (RL)
1JourA	M1	2083	2085	1	5693 (5018)	14527 (14511)	224973 (78351)	**8** (2)
	M2	2081	2085	1	6333 (5574)	14527 (14524)	246153 (87275)	27 (5)
	M3	2080	2085	1	5025 (5021)	14527 (14522)	64214 (63818)	20 (2)
	M4	2082	2085	1	5665 (5572)	14527 (14521)	85394 (75812)	21 (3)
	M3+F2	2085	2085	1	2410 (2088)	5915 (5480)	22721 (21726)	2 (0)
1JourB	M1	2099	2130	109	6989 (6370)	21592 (21585)	333100 (120357)	**91** (6)
	M2	2101	2130	223	7419 (6776)	21592 (21589)	375203 (140065)	258 (17)
	M3	2098	2130	589	6377 (6373)	21592 (21588)	103418 (102466)	175 (5)
	M4	2099	2130	333	6807 (6775)	21592 (21588)	145521 (122562)	336 (10)
	M3+F2	2100	2130	69	3323 (3095)	9417 (9090)	40520 (39707)	31 (1)
1JourC	M1	7005	7035	289	6801 (6157)	19877 (19869)	307387 (109500)	186 (8)
	M2	7005	7035	101	7269 (6573)	19877 (19873)	345857 (126875)	149 (13)
	M3	7007	7035	119	6165 (6159)	19877 (19871)	93516 (92766)	**113** (7)
	M4	7005	7035	195	6633 (6571)	19877 (19871)	131986 (11107)	247 (14)
1JourD	M1	62770	62805	771	9516 (9086)	50187 (49701)	849829 (285627)	1524 (32)
	M2	62777	62805	185	11051 (10562)	50187 (49749)	1014825 (370516)	1545 (84)
	M3	62701	62805	483	9088 (9081)	50187 (49604)	246794 (242803)	**1073** (43)
	M4	62707	62805	195	10623 (10557)	50187 (49616)	411790 (327574)	1310 (89)
1JourE	M1	44562	44580	2269	10348 (9912)	55472 (55073)	948809 (320679)	1400 (41)
	M2	44562	44580	1895	12012 (11536)	55472 (55092)	1158999 (427458)	5125 (124)
	M3	44562	44580	721	9912 (9906)	55472 (54970)	275732 (272318)	**796** (38)
	M4	44562	44580	1058	11576 (11530)	55472 (54970)	485922 (378974)	4115 (112)
2JoursA	M1	25019	25560	51	15912 (14321)	38083 (37986)	580461 (221040)	**107** (16)
	M2	24885	25560	83	17182 (15586)	38083 (38058)	681962 (262797)	514 (33)
	M3	24988	25560	105	14328 (14325)	38083 (38023)	166538 (164914)	216 (16)
	M4	24927	25560	81	15598 (15589)	38083 (38068)	268039 (228884)	314 (40)
2JoursB	M1	4078	4110	4833	18864 (17676)	66325 (66210)	989688 (366500)	816 (24)
	M2	4078	4110	759	20134 (18938)	66325 (66304)	1214527 (468225)	1707 (51)
	M3	4082	4110	365	17676 (17676)	66325 (66261)	315113 (312498)	**261** (20)
	M4	4078	4110	587	18946 (18938)	66325 (66313)	539952 (414924)	823 (47)
	M3+F2	4083	4110	831	6354 (5868)	17948 (17234)	72602 (70848)	152 (2)
2JoursC	M1	52577	53535	315	15495 (13901)	36774 (36666)	562675 (212386)	**433** (20)
	M2	52436	53535	847	16937 (145321)	36774 (36749)	656578 (247925)	1752 (42)
	M3	52302	53535	701	13911 (13908)	36774 (36714)	158348 (156756)	639 (19)
	M4	52329	53535	713	15353 (15324)	36774 (36759)	252251 (218431)	1581 (50)
2JoursD	M1	170100	170175 (170140)	3899	16669 (15794)	81250 (80334)	1342897 (461901)	7200 (62)
	M2	170098	170175 (170137)	1031	20191 (19217)	81250 (80694)	1597154 (594518)	7200 (214)
	M3	170092	170175 (170136)	3634	15809 (15800)	81250 (80549)	392730 (387348)	7200 (68)
	M4	170101	170175 (170133)	929	19331 (19210)	81250 (80636)	646987 (518901)	7200 (250)
2JoursE	M1	143417	143730	8913	15724 (14737)	72675 (71523)	1250028 (376617)	7008 (47)
	M2	143329	143745 (143682)	1834	19096 (17946)	72675 (71946)	1464543 (485275)	7200 (134)
	M3	143591	143730	7803	14784 (14745)	72675 (71515)	334463 (326836)	**6772** (53)
	M3	143417	143760 (143676)	7803	18156 (17915)	72675 (71915)	548978 (436310)	7200 (133)
7JoursA	M1	11929	11940	1	64712 (60071)	211137 (211095)	3600043 (1324545)	426 (75)
	M2	11921	11940	75	69210 (64569)	211137 (211134)	4589911 (1694720)	2111 (138)
	M3	11912	11940	69	60072 (60072)	211137 (211120)	1028160 (1022510)	**323** (49)
	M4	11918	11940	217	64570 (64570)	211137 (211135)	2018028 (1516327)	2278 (126)
	M3+F2	11934	11940	1	19178 (18203)	51127 (49938)	203044 (200045)	38 (5)
7JoursB	M1	11473	11490 (11474)	9365	61928 (57283)	197837 (197697)	3368648 (1219035)	7200 (72)
	M2	11463	11490	665	67318 (62677)	197837 (197814)	4267782 (1568142)	**4058** (159)
	M3	11455	11490	8735	57288 (57284)	197837 (197769)	936718 (930853)	4875 (47)
	M4	11457	11490 (11470)	1558	62678 (62678)	197837 (197827)	1835852 (1403737)	7200 (118)
	M3+F2	11468	11490	505	18451 (17290)	50270 (48790)	198543 (194911)	281 (5)
7JoursC	M1	33298	33330	399	63592 (58951)	207573 (207531)	3551226 (1286996)	1367 (88)
	M2	33294	33330 (33310)	784	68650 (64009)	207573 (207570)	4503740 (1666560)	7200 (208)
	M3	33298	33330	107	58952 (58952)	207573 (207556)	1006370 (1000758)	**632** (72)
	M4	33296	33330 (33305)	748	64010 (64010)	207573 (207571)	1958884 (1483534)	7200 (211)
7JoursD	M1	674462	701640 (674946)	94	57479 (54542)	301891 (300508)	5103108 (1654750)	7200 (490)
	M2	673562	— (674068)	60	73737 (70420)	301891 (301120)	5928199 (2123525)	7200 (1570)
	M3	672862	— (673710)	179	54567 (54524)	301891 (300721)	1420547 (1408233)	7200 (541)
	M4	672743	— (673576)	47	70825 (70381)	301891 (300899)	2245638 (1876470)	7200 (1628)
7JoursE	M1	704605	— (705473)	83	55546 (52406)	276269 (274855)	4600899 (1513185)	7200 (520)
	M2	705534	— (706258)	78	71833 (68155)	276269 (275891)	5351323 (1931114)	7200 (1502)
	M3	705033	— (705570)	155	52470 (52404)	276269 (275194)	1293443 (1280185)	7200 (548)
	M3	705821	— (706291)	68	68757 (68088)	276269 (275404)	2043867 (1695937)	7200 (1676)

demandes réelles. Un exemple tiré de l'instance 1jourD est une demande de 3 suivi d'une demande de 6 suivi d'une demande de 3 sur trois périodes pour une même activité qui a une durée minimale de 6 périodes.

Ce qui apparaît cependant plus clairement, ce sont les mérites des différentes approches. Afin d'illustrer plus clairement ces comparaisons, le tableau récapitulatif 6.4 présente le modèle, le nombre de meilleur rang, le nombre de solutions entières obtenues, le nombre de solutions optimales obtenues, le rang pour chaque instance et le rang moyen. Les rangs sont déterminés par le temps de calcul pour trouver la solution optimale ou la comparaison des bornes inférieures pour les méthodes n'ayant pas trouvé la solution optimale. Quoique les modèles **M3** et **M4** basés sur une formulation avec contraintes de flot ont moins de rangées et beaucoup moins d'éléments non nuls dans la formulation de départ que les modèles **M1** et **M2** correspondants, cet avantage est grandement diminué après le pré-traitement. Néanmoins, les formulations basées sur des contraintes de flot conservent un avantage significatif en ce qui à trait au nombre d'éléments non nuls. Cependant cet avantage ne se traduit pas en une efficacité générale accrue. Les méthodes basées sur les modèles **M1** et **M3** livrent des performances semblables, tout comme les méthodes basées sur les modèles **M2** et **M4**. Il aurait été possible que la formulation de flot ait nui à la génération de coupes, mais nos résultats ne pointent pas dans cette direction et nous ne trouvons pas d'explications satisfaisantes à ces résultats. Cependant il apparaît clairement, que en dépit d'une relaxation linéaire moins serrée (quoique en pratique la différence est infime), les méthodes basées sur les modèles **M1** et **M3** sont beaucoup plus performantes que les méthodes basées sur les modèles **M2** et **M4**. Dans ce cas, la différence en nombre d'éléments non nuls n'est pas affectée par le pré-traitement, et, quoique cette différence en nombre d'éléments non nuls est semblable à cette même différence entre les modèles **M3** et **M4** et les modèles **M1** et **M2**, le nombre d'éléments non nuls semble cette fois avoir de l'importance.

Tableau 6.4 – Tableau récapitulatif des résultats pour les modèles de PLNE.

Modèle	Nb de meilleurs rangs	Nb de solutions optimales	Nb de solutions entières	Rang	Rang moyen
M1	6	11	14	1, 1, 3, 3, 2, 1, 2, 1, 1, 2, 2, 3, 2, 1, 4	1.93
M2	1	10	13	4, 3, 2, 4, 4, 4, 4, 4, 2, 3, 3, 1, 3, 2, 2	3
M3	7	12	13	2, 2, 1, 1, 1, 2, 1, 2, 3, 1, 1, 2, 1, 3, 3	1.73
M4	1	9	13	3, 4, 4, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 1	3.33

Finalement, il apparait clairement que la méthode de réduction $F2$ a un impact considérable sur la rapidité d'exécution : les temps de calcul de cette méthode sont toujours inférieurs au meilleur temps de calcul des autres (en moyenne 6.32 plus rapide que le meilleur temps). Quoique cette méthode a une portée limitée car restreinte aux problèmes dont on sait qu'il n'existe pas de sous/sur-couvertures, elle est suffisamment performante pour justifier son emploi dans un contexte où l'on voudrait d'abord chercher une solution sans sous/sur-couvertures à l'aide de cette méthode. Si il n'y a pas de telle solution, on pourrait utiliser la méthode générale pour trouver la solution optimale tout en ajoutant une contrainte pour forcer le nombre de variables de sous-couvertures (et aussi de sur-couvertures) à être plus grand ou égal à 1 et ainsi peut-être augmenter les bornes inférieures obtenues par la relaxation linéaire. Cette approche serait d'autant plus justifiée, que dans la plupart des cas testés, quand il existe au moins une sous-couverture, la relaxation linéaire est non réalisable et la méthode utilisant $F2$ se termine rapidement, le temps pris par le pré-traitement serait alors très petit. Il est aussi envisageable de modifier la technique afin de généraliser son utilisation. Notons pour fin de comparaison, que les techniques utilisées pour générer nos instances utilisent des valeurs cibles qui visent à la création de problèmes similaires à des problèmes d'horaires issus de situations réelles de planification d'horaires pour les contrôleurs aériens qui sont difficiles à résoudre avec les méthodes existantes. Dans ce contexte, les résultats obtenus sont très prometteurs.

6.6 Conclusion

Cette recherche visait à obtenir une technique de résolution efficace pour le problème d'assignation d'activités à des quarts de travail. Pour ce faire, nous avons d'abord élaboré une approche métaheuristique basée sur une méthode tabou et utilisant des problèmes de flot pour réduire la taille du problème. Cette méthode s'est montrée d'une efficacité limitée. Elle a cependant mis en lumière l'efficacité des méthodes de réduction et inspiré une modélisation efficace du problème comme un PLNE. Nous avons développé une méthode de résolution basée sur quatres modèles de programmation linéaire en nombres entiers et sur l'utilisation des techniques de réduction développées pour la méthode tabou. Un logiciel commercial a été utilisé pour résoudre ces modèles et les résultats obtenus sont excellents. La comparaison des modèles a permis d'en établir deux comme clairement supérieurs, mais, malgré des considérations théoriques laissant penser qu'un de ces deux modèles serait supérieur à l'autre, ces deux modèles se sont avérés aussi efficaces l'un que l'autre et possiblement complémentaires. Une des techniques de réduction, applicable uniquement sur certains types de problèmes s'est aussi révélée très efficace. Ces résultats nous portent à croire que la méthode de résolution basée sur un PLNE que nous avons établie est très prometteuse car elle est efficace et, par sa simplicité, offre la possibilité de nombreuses améliorations.

DISCUSSION GÉNÉRALE ET CONCLUSION

Les problèmes de fabrication d'horaires sont issus de plusieurs secteurs différents et sont, par conséquent, nombreux et variés. C'est dans ce domaine que nous avons puisé le sujet de cette thèse. Nous nous sommes d'abord interressé à la résolution de problèmes de coloration des arêtes et des nœuds d'un graphe. Ces problèmes de coloration permettent de modéliser certaines classes de problèmes d'horaires. Nous avons donné une attention particulière à la coloration de graphe modélisant des horaires qui présentent des contraintes de compacité (i.e. des contraintes exigeant la consécutivité de certaines périodes).

Dans cette thèse, nous avons d'abord présenté, pour la première fois, un algorithme permettant de résoudre le problème de minimisation de la déficience pour un graphe général. L'algorithme développé, basé sur la méthode tabou s'est révélé beaucoup plus efficace qu'un autre algorithme basé sur une implémentation plus simple de la méthode tabou. L'idée principale de la méthode proposée reposait sur l'utilisation d'un opérateur d'échange bi-chromatique pour générer des voisinages de meilleure qualité. Nous avons aussi proposé une borne inférieure sur le nombre de couleurs nécessaires si l'on souhaite obtenir une coloration ayant une déficience donnée. Cette information a permis de mettre en lumière la raison pour laquelle l'algorithme avait de la difficulté avec certains graphes (ceux dont les colorations optimales nécessitent substantiellement plus de couleurs que leur indice chromatique) et d'apporter des correctifs. Il reste toutefois que la méthode tabou, telle que nous l'avons implantée, a tendance, naturellement, à visiter des solutions utilisant peu de couleurs. Un défi intéressant pour une recherche future serait de trouver une façon plus efficace de favoriser l'exploration de coloration utilisant plus de couleurs.

Ensuite, notre intérêt s'est porté sur le problème de coloration par intervalle. Dans un

premier temps nous avons mis en lumière la différence entre celui-ci et un problème semblable, la coloration par bande. Malgré cette différence, nous avons démontré qu'il est possible de réduire le problème de coloration par intervalle à une série de problèmes de coloration par bande. Cette réduction a été mise en pratique dans un algorithme utilisant des méthodes connues pour résoudre la série de problèmes de coloration par bande. Les résultats obtenus ont montré que l'algorithme se compare avantageusement à un algorithme existant utilisé à titre de comparaison. Un gain de 3% a été observé sur les graphes du banc de test DIMACS. De plus, il permet, dans sa version exacte et dans sa version heuristique, d'obtenir de meilleures bornes supérieures sur le nombre maximal de couleurs nécessaires pour la coloration d'instances difficiles. Finalement, dans le cas où une borne inférieure est fournie et où l'on veut vérifier si une coloration atteint cette borne, l'agorithme est particulièrement efficace car la réduction se résume à un seul problème de coloration par bande. Dans ces cas, ce dernier a été capable de résoudre de plus grandes instances que l'algorithme utilisé pour fin de comparaison.

Nous avons poursuivi nos recherches sur le problème de coloration par intervalle en nous attaquant à un enjeu, toujours important dans la résolution de problèmes d'optimisation combinatoire, qui est d'identifier des symétries dans les solutions afin de réduire l'espace des solutions. Dans le contexte de la coloration par intervalle, cet objectif consiste, entre autres choses, à identifier si une coloration n'est en fait que la permutation des couleurs d'une autre coloration. Si c'est le cas, les deux colorations sont, à toute fin pratique, équivalentes, et il suffit d'en évaluer juste une lors de l'exploration de l'espace des solutions. Dans ce but, nous avons proposé une condition nécessaire et suffisante pour que deux colorations soient équivalentes. Nous avons intégré cette condition au sein d'une méthode tabou afin d'illustrer son utilité. Quoique les résultats obtenus soient suffisants pour justifier la pertinence de la condition d'équivalence, nous croyons qu'une utilisation plus judicieuse de cette

dernière reste à trouver pour la méthode tabou. Il serait aussi intéressant d'évaluer comment cette condition pourrait être intégrée à d'autres techniques de résolution.

Le dernier problème abordé dans cette thèse est celui de l'attribution d'activités à des quarts de travail pour lequel nous cherchions à obtenir une technique de résolution efficace. Les techniques connues précédemment utilisées pour résoudre ce problème sont toutes basées sur l'utilisation de la génération de colonnes combinée à une approche de séparation et d'évaluation progressive. Mais, en raison de la taille des problèmes, un élément heuristique est toujours ajouté à la méthode pour en accélérer la vitesse d'exécution. Dans ce contexte, nous avons d'abord essayé de résoudre ce problème en utilisant une approche purement heuristique basée sur une méthode tabou. Une technique de réduction de la taille du problème utilisant deux problèmes de flot a aussi été développée. L'approche heuristique, quoique convenable, n'a pas été à la hauteur de nos attentes ; elle a cependant mis en lumière l'efficacité des méthodes de réduction et inspiré une modélisation efficace du problème comme un programme linéaire en nombres entiers. Nous avons mis à l'essai plusieurs variantes de ce modèle de programmation linéaire en nombres entiers et deux de celles-ci se sont avérées très efficaces. Les expérimentations numériques ont aussi montré l'efficacité des méthodes de réduction de la taille du problème tant pour l'approche heuristique que pour l'approche exacte. Certaines questions restent en suspens quant à l'efficacité de certaines variantes du modèle. Dans un développement futur, nous croyons que le domaine d'application des techniques de réduction pourrait être étendu.

L'objectif de cette thèse était de développer de nouvelles méthodes de résolution pour des problèmes de fabrication d'horaires ou pour des problèmes qui leur sont connexes. Pour chacun des problèmes considérés, une approche nouvelle a été développée. Ces approches sont fondées sur des considérations théoriques que nous avons établies et qui permettent d'évaluer leurs forces et leurs faiblesses. Les méthodes présentées se sont révélées efficaces et susceptibles de mener à des développements ultérieurs.

BIBLIOGRAPHIE

ADDOU, I. (2005). *Généralisation aux extra-chevauchements du modèle de Bechtold-Jacobs pour les horaires de personel.* Mémoire de maîtrise, École Polytechnique, Montréal, Canada.

ASRATIAN, A.S. et CASSELGREN, C.J. (2006). On interval edge colorings of (α, β)-biregular bipartite graphs. *Discrete Mathematics* **307**, 1951-1956.

ASRATIAN, A.S. et DE WERRA, D. (2002). A generalized class-teacher model for some timetabling problems. *Discrete Optimization* **143**, 531-542.

ASRATIAN, A.S. et KAMALIAN R.R. (1994). Investigation on interval edge-colorings of graphs. *Journal of Combinatorial Theory* **62**, 34-43.

AYKIN, T. (1996). Optimal shift scheduling with multiple break windows. *Management Science* **42**, 591-602.

AYKIN, T. (1998). A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *Journal of the Operational Research Society* **49**, 603-615.

AYKIN, T. (2000). A comparative evaluation of modeling approaches to the labor shift scheduling problem. *European Journal of Operational Research* **125**, 381-397.

BAILEY, J. et FIELDS, J. (1985). Personnel scheduling with flexshift models. *Journal of Operations Management* **5**, 327-338.

BARD, J.F., BINICI, C. et DE SILVA, A.H. (2003). Staff scheduling at the United States postal service. *Computers and Operations Research* **30**, 745-771.

BARTHOLDI, J.J. et MCCROAN K.L. (1990). Scheduling interviews for a job fair. *Operations Research* **38**, 951-960.

BECHTOLD, S.E. et JACOBS, L.W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science* **44**, 534-547.

BECHTOLD, S.E. et JACOBS, L.W. (1996). The equivalence of general set-covering and implicit integer programming formulations for shift scheduling. *Naval Research Logistics* **43**, 233-249.

BIGRAS, L.P., GAMACHE, M. et SAVARD G. (2008). *Time-indexed formulations and the total weighted tardiness problem.* INFORMS Journal on Computing 20, 133-142.

BOUCHARD, M. (2004). *Optimisation des pauses dans le problème de fabrication d'horaires avec quarts de travail.* Mémoire de maîtrise, École Polytechnique, Montréal, Canada.

BRUSCO, M.J. et JACOBS, L.W. (1998). Personnel tour scheduling when starting-time restrictions are present. *Management Science* **44**, 534-547.

BURNS, R.N. et CARTER, M.W. (1985). Workforce size and single shift schedules with variable demands. *Management Science* **31**, 599-607.

BURNS, R.N. et KOOP, G.J. (1987). A modular approach to optimal multiple-shift manpower scheduling. *Operations Research* **35**, 100-110.

ÇEZIK, T. et GÜNLÜK, O. (2004). Reformulating linear programs with transportation constraints - With applications to workforce scheduling. *Naval Research Logistics* **51**, 275-296.

CLEMENTSON, A.T. et ELPHICK, C.H. (1983). Approximate colouring algorithms for composite graphs. *Journal of Operational Research Society* **34**, 503-509.

DANTZIG, G.B. (1954). A comment on Edie's traffic delays at toll booths. *Operations Research* **2**, 339-341.

DE WERRA, D. (1990). Almost nonpreemptive schedules. *Annals of Operations Research* **26**, 243-256.

DE WERRA, D. (1997). Restricted coloring models for timetabling. *Discrete Mathematics* **165/166**, 161-170.

DE WERRA, D. (1999). Restricted graph coloring : some mathematical programming models. *CRM Proceedings and Lecture Notes* **23**, 135-148.

DE WERRA, D., ASRATIAN, A.S. et DURAND, S. (2002). Complexity of some special types of timetabling problems. *Journal of Scheduling* **5**, 171-183.

DE WERRA, D., BLAZEWICZ, J. et KUBIAK, W. (1991). A preemptive open shop scheduling problem with one resource. *Operations Research Letter* **10**, 9-15.

DE WERRA, D. et HERTZ, A. (1988). Consecutive colorings of graphs. *Zeitschrift für Operations Research* **32**, 1-8.

DE WERRA, D. et SOLOT, P. (1991). Compact cylindrical chromatic scheduling. *SIAM Journal on Discrete Mathematics* **49**, 603-615.

DE WERRA, D. et SOLOT, P. (1993). Some graph-theoretical models for scheduling in automated production systems. *Networks* **23**, 651-660.

DEUBER, W.A et ZHU, X. (1996). Circular colorings of weighted graphs. *Journal of Graph Theory* **24**, 365-376.

EASTON, F.F. et ROSSIN, D.F. (1991). Sufficient working subsets for the tour scheduling problem. *Management Science* **37**, 1441-1451.

EDIE, L. C. (1954). Traffic delays at toll booths. *Operations Research* **2**, 107-137.

EMMONS, H. et BURNS, R.N. (1991). Off-day scheduling with hierarchical worker categories. *Operations Research* **39**, 484-496.

GABOW, H.N. (1976). Using Euler partitions to edge color bipartite multigraphs. *International Journal of Computer and Information Sciences* **5**, 345-354.

GABOW, H.N. et KARIV, O. (1982). Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal on Computing* **11**, 117-129.

GIARO, K. (1997). The complexity of consecutive Δ-coloring of bipartite graphs : 4 is easy, 5 is hard. *Ars Combinatoria* **47**, 287-298.

GIARO, K. (2001). NP-hardness of compact scheduling in simplified open and flow shops. *European Journal of Operational Research* **130**, 90-98.

GIARO, K., KUBALE, M. et MALAFIEJSKI, M. (1999). On the deficiency of bipartite graphs. *Discrete Applied Mathematics* **94**, 193-203.

GIARO, K., KUBALE, M. et MALAFIEJSKI, M. (2001). Consecutive colorings of the edges of general graphs. *Discrete Mathematics* **236**, 131-143.

GIARO, K. et KUBALE, M. (2004). Compact scheduling of zero-one time operations in multi-stage systems. *Discrete Applied Mathematics* **145**, 95-103.

GIBBONS, A. (1985). Algorithmic graph theory. Cambridge University Press.

GLOVER, F. et LAGUNA, M. (1997). Tabu Search. Kluwer Academic Publishers, Boston.

HAASE, K. (1999). Advanced column generation techniques with application to marketing, retail and logistics management. Post-doctoral thesis, University of Kiel.

HANSON, D. LOTEN, C.O.M. et TOFT, B. (1996). A lower bound for interval colouring of bi-regular bipartite graphs. *Bulletin of the Institute of Combinatorics and Applications* **18**, 69-74.

HANSON, D. LOTEN, C.O.M. et TOFT, B. (1998). On interval colorings of bi-regular bipartite graphs. *Ars Combinatoria* **50**, 23-32.

HENDERSON, W.B. et BERRY, W.B. (1979). Heuristic methods for telephone operator shift scheduling : an experimental analysis. *Management Science* **22**, 1372-1380.

HERTZ, A. et DE WERRA, D. (1987). Using tabu search techniques for graph coloring. *Computing* **39**, 345-351.

IRNICH, S. et DESAULNIERS, G. (2005). Shortest path problems with resource constraints. *Column Generation, G. Desaulniers, J. Desrosiers and M.M. Solomon (eds)*, Springer, NY, 33-65.

TOFT, B. et JENSEN, T.R. (1995). Graph Coloring Problems. Wiley, New York.

HOLYER, I. (1981). The $\mathcal{NP}$-completeness of edge-coloring. *SIAM Journal on Computing*, **4**, 718-720.

JARRAH, A.I.Z., BARD, J.F. et DESILVA, A.H. (1994). Solving large-scale tour scheduling problems. *Management Science* **40**, 1124-1144.

KEITH, E.G. (1979). Operator scheduling. *AIIE Transactions* **11**, 37-41.

KUBALE, M. (1989). Interval vertex-coloring of a graph with forbidden colors. *Discrete Mathematics* **74**, 125-136.

KUBALE, M. (1992). Some results concerning the complexity of restricted colorings of graphs. *Discrete Applied Mathematics* **36**, 35-46.

KUBALE, M. et NADOLSKI, A. (2005). Chromatic scheduling in a cyclic open shop. *European Journal of Operational Research* **164**, 585-591.

MOONDRA S.L. (1976). An L.P. model for work force scheduling in banks. *Journal of Bank Research* **6**, 299-301.

OMARI, Z. (2002). *Attribution des activités aux employés travaillant sur des quarts.* Mémoire de maîtrise, École Polytechnique, Montréal, Canada.

PRESTWICH, S. (2002). Constrained bandwidth multicoloration neighborhoods. *Proceedings of Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, NY, USA, 126-133.

PUNTER, A. (2004). *Systems for timetabling by computer based on graph coloring.* Ph.D. Thesis, C.N.A.A., Hatfield Polytechnic, 1976.

PYATKIN, A.V. (2004). Interval coloring of $(3, 4)$-biregular bipartite graphs having large cubic subgraphs. *Journal of Graph Theory* **47(2)**, 122-128.

REKIK, M., CORDEAU, J.F. et SOUMIS, F. (2004). Using Benders decomposition to implicitly model tour scheduling. *Annals of Operations Research* **128**, 111-133.

SCHRIJVER, A. (1964). Bipartite edge coloring in $O(\Delta m)$ time. *SIAM Journal on Computing* **28(3)**, 841-846.

SEGAL, M. (1974). The operator scheduling problem : a network flow approach. *Operations Research*, **22**, 808-824.

SEVASTJANOW, S.V. (1990). On interval edge colouring of bipartite graphs (in Russian). *Metody Diskret Analiz* **50**, 61-72.

THOMPSON, G.M. (1995). Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science* **41**, 595-607.

TOPALOGLU, G.M. et OZKARAHAN, I. (2002). A goal programming approach for large-scale tour scheduling problems with flexible break assignments. *Proceedings of the 31st Annual Meeting of the Decision Science Institute, Orlando, Florida.*

VATRI, E. (2001). *Intégration de la génération de quarts de travail et de l'attribution d'activités.* Mémoire de maîtrise, École Polytechnique, Montréal, Canada.

VIZING, V.G. (1965). The chromatic index of a multigraph. *Kibernetika.* **3**, 29-39.