Titre: Title:	Complexité et cassage de symétrie pour le problème de la déficience d'un graphe
Auteur: Author:	Sivan Altinakar
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Altinakar, S. (2016). Complexité et cassage de symétrie pour le problème de la déficience d'un graphe [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2330/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2330/
Directeurs de recherche: Advisors:	Alain Hertz, & Gilles Caporossi
Programme: Program:	Mathématiques de l'ingénieur

UNIVERSITÉ DE MONTRÉAL

COMPLEXITÉ ET CASSAGE DE SYMÉTRIE POUR LE PROBLÈME DE LA DÉFICIENCE D'UN GRAPHE

SIVAN ALTINAKAR
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

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

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

COMPLEXITÉ ET CASSAGE DE SYMÉTRIE POUR LE PROBLÈME DE LA DÉFICIENCE D'UN GRAPHE

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

M. SOUMIS François, Ph. D., président
M. HERTZ Alain, Doct. ès Sc., membre et directeur de recherche
M. CAPOROSSI Gilles, Ph. D., membre et codirecteur de recherche
Mme MARCOTTE Odile, Ph. D., membre
M. KUBIAK Wieslaw, Ph. D., membre externe

DÉDICACE

À tous ceux qui dansent en marchant !

REMERCIEMENTS

À toutes les personnes qui ont fait partie de ma vie lors de ce voyage initiatique qu'est une thèse, je tiens à personnellement témoigner...

Ma reconnaissance sans bornes à mes deux co-directeurs Alain Hertz et Gilles Caporossi, qui n'ont jamais failli à me soutenir dans l'inconnu et les moments de contraintes, et dont la patience à mon égard mériterait d'être racontée dans une épopée.

Mes remerciements aux membres de mon jury Wieslaw Kubiak, Odile Marcotte et François Soumis, pour avoir accepté de faire partie de mon jury de thèse.

Mon affection émue à mes parents, qui m'ont encouragé dans cette voie, et qui ont généreusement choisi le difficile chemin de me pousser au-delà de ce que je pensais être capable, et ont accepté d'en assumer les contraignantes conséquences.

Mes belles pensées à Deborah Ummel, avec qui je me suis lancé dans cette aventure. . . et sans qui je ne l'aurais jamais terminée.

RÉSUMÉ

Une coloration d'arête d'un graphe $G = (V, E)$ est une fonction c qui assigne un entier $c(e)$ (appelé une couleur) dans $\{0, 1, 2, \ldots\}$ à chaque arête $e \in E$ de sorte que des couleurs différentes soient assignées à des arêtes adjacentes. Une coloration d'arête est compacte si les couleurs des arêtes incidentes à chaque sommet forment un ensemble d'entiers consécutifs. Le problème appelé déficience consiste à déterminer le nombre minimum d'arêtes pendantes à rajouter au graphe pour que le graphe résultant ait une coloration d'arête compacte. Parmi les variations de ce problème, on compte le problème de la coloration d'arête compacte linéaire ($k - LCCP$) où il est possible d'utiliser uniquement les k couleurs dans $\{0, 1, \ldots, k-1\}$, et le problème de la coloration d'arête compacte cyclique ($k - CCCP$) où additionnellement la couleur 0 est considérée consécutive à la couleur $k - 1$.

Nous proposons une réduction polynomiale du $k - LCCP$ (optionnellement avec des couleurs imposées ou interdites sur certaines arêtes) au $k - CCCP$ lorsque $k \geq 12$, et au 12-CCCP lorsque $k < 12$. Nous proposons et comparons également la performance de 3 modélisations en Programmation en Nombres Entiers et un modèle en Programmation par Contraintes pour le problème de la déficience, et déterminons le dernier comme étant le plus approprié pour ce problème.

En raison des symétries, une instance du problème de déficience peut avoir de nombreuses solutions optimales équivalentes. Nous présentons une approche pour générer un petit ensemble de contraintes, appelées GAMBLLE, destinée à casser la symétrie, qui peuvent être incorporées au modèle en programmation par contrainte. Les contraintes GAMBLLE sont inspirées des contraintes de Lex-Leader, basées sur les automorphismes de graphe, et agissent sur des familles de variables permutables. Nous analysons leur impact sur la réduction du nombre de solutions optimales, ainsi que le gain de temps obtenu lors de la résolution d'une modélisation en programmation par contrainte.

ABSTRACT

An edge-coloring of a graph $G = (V, E)$ is a function c that assigns an integer $c(e)$ (called color) in $\{0, 1, 2, \ldots\}$ to every edge $e \in E$ so that adjacent edges are assigned different colors. An edge-coloring is compact if the colors of the edges incident to every vertex form a set of consecutive integers. The deficiency problem is to determine the minimum number of pendant edges that must be added to a graph such that the resulting graph admits a compact edge-coloring. Variations of this problem include the linear compact k-edge-coloring problem ($k - LCCP$) where there are only the k colors of $\{0, 1, \ldots, k-1\}$ available, and the cyclic compact k-edge-coloring problem ($k - CCCP$) where additionally color 0 is considered consecutive to color $k - 1$.

We demonstrate a polynomial reduction of the $k - LCCP$ (with optionally additional imposed or forbidden colors on some edges) to the $k - CCCP$ when $k \geq 12$, and to the $12 - CCCP$ when $k < 12$. We also propose and compare the performance of three integer programming models and one constraint programming model for the deficiency problem, and determine the latter to be the best suited to model this problem.

Because of symmetries, an instance of the deficiency problem can have many equivalent optimal solutions. We present a way to generate a small set of symmetry breaking constraints, called GAMBLLE constraints, that can be added to a constraint programming model. The GAMBLLE constraints are inspired by the Lex-Leader ones, based on automorphisms of graphs, and act on families of permutable variables. We analyze their impact on the reduction of the number of optimal solutions as well as on the speed-up of the constraint programming model.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

LCCP	Linear Compact (edge-)Coloring Problem
CCCP	Cyclic Compact (edge-)Coloring Problem
k-**LCCP**	Linear Compact k-(edge-)Coloring Problem
k-**CCCP**	Cyclic Compact k-(edge-)Coloring Problem
GAMBLLE	Graph AutoMorphism-Based Lex-Leader Enforcing (constraint)

CHAPITRE 1 INTRODUCTION

La déficience, définie dans le cadre de la théorie des graphes, est un invariant de graphe. Son origine provient d'un problème de confection d'horaire où certains agents doivent avoir un horaire le plus compact possible, modélisé sous la forme d'un problème de coloration d'arêtes. Il existe en particulier une variation intéressante où un horaire est amené à se répéter de manière périodique, qui peut être comparé au problème original. Par ailleurs, il n'existe pas d'algorithme exact efficace pour résoudre ce problème, même pour des petits graphes. Cette difficulté provient en partie d'un grand nombre de solutions optimales équivalentes, engendrées par les automorphismes du graphe. Pour faciliter le travail d'un solveur, il est possible, par exemple, de rajouter des contraintes destinées à briser cette symétrie.

1.1 Définitions et concepts de base

Tous les graphes considérés n'ont pas de boucle, et peuvent avoir des arêtes multiples. Une *k-coloration d'arête* d'un graphe $G = (V, E)$ est une fonction $c : E \to \{0, 1, \cdots, k-1\}$ qui assigne une couleur $c(e)$ à toute arête $e \in E$ telle que $c(e) \neq c(e')$ si e et e' partagent une même extrêmité. Soit E_v l'ensemble des arêtes incidentes au sommet $v \in V$. Le *degré* $\deg(v) = |E_v|$ de v est le nombre d'arêtes incidentes à v, et le degré maximum présent dans G est noté $\Delta(G)$. Toute coloration d'arêtes d'un graphe G utilise au moins $\Delta(G)$ couleurs, et donc $k \geq \Delta(G)$.

Une k-coloration d'arêtes d'un graphe $G = (V, E)$ est *(linéaire) compacte* si $\{c(e) : e \in E_v\}$ est un ensemble d'entiers consécutifs pour tout sommet $v \in V$. Un graphe est *linéaire compact colorable* s'il admet une k-coloration d'arêtes linéaire compacte pour un certain entier k. Soit une k-coloration d'arêtes c de G, $c_{min}(v) = \min_{e \in E_v}\{c(e)\}$ et $c_{max}(v) = \max_{e \in E_v}\{c(e)\}$ définissent respectivement la plus petite et la plus grande couleur assignée à une arête incidente à v. Si c est linéaire compacte, ces dernières sont liées par $c_{max}(v) = c_{min}(v) + \deg(v) - 1$ pour tout sommet $v \in V$. Le problème consistant à déterminer si G est k-colorable linéaire compact est noté k-LCCP (pour Linear Compact k-Coloring Problem).

Une k-coloration d'arêtes est *cyclique compacte* s'il existe deux entiers $a_v, b_v < k$ pour chaque sommet v tel que $\{c(e) : e \in E_v\} = \{a_v, (a_v + 1) \bmod k, \cdots, (a_v + |E_v| - 1) \bmod k = b_v\}$ (i.e. la couleur 0 est considérée consécutive à $k - 1$). Un graphe est *cyclique compact colorable* s'il admet une k-coloration d'arêtes compacte pour un entier k. Le problème consistant à déterminer si G est k-colorable cyclique compact est noté k-CCCP (pour Cyclic Compact

k-Coloring Problem).

Par exemple, le *open shop problem* peut facilement se modéliser en ces termes. Il se base sur m processeurs $P_1, \cdots, P_m$ et n jobs $J_1, \cdots, J_n$. Chaque job J_i est un ensemble s_i de tâches. Supposons que chaque tâche doit être traitée en une unité de temps sur un processeur spécifique. Deux tâches du même job ne peuvent être traitées simultanément, et un processeur ne peut travailler sur deux tâches en même temps. Une contrainte de compacité requiert qu'aucun job n'ait de temps d'attente entre ses tâches, et qu'aucun processeur n'ait de temps d'inutilisation entre ses tâches. En d'autres mots, les périodes de temps affectées aux tâches d'un job doivent être consécutives, et chaque processeur doit être actif pendant un ensemble consécutif de périodes. L'existence d'un horaire réalisable compact sur k périodes de temps est équivalent à déterminer l'existence d'une k-coloration d'arêtes linéaire compacte du graphe G qui contient un sommet pour chaque job et chaque processeur, et une arête pour chaque tâche (i.e. une tâche d'un job J_i à traiter sur P_j est représentée par une arête entre les sommets représentant J_i et P_j). Chaque couleur utilisée dans la k-coloration correspond à une période de temps. La contrainte de compacité pour chaque job et processeur est équivalente à imposer que les couleurs affectées aux arêtes de E_v doivent être consécutives pour chaque sommet v de G. Dans nombre de systèmes de production automatisée, la production est organisée d'une manière cyclique, i.e. le même horaire de production de durée T doit être répété en continu à chaque T unité de temps. La contrainte de compacité impose alors que les périodes de temps assignées aux tâches de chaque job et les périodes d'activité de chaque processeur forment des intervalles cycliques dans chaque cycle de production. L'existence d'un horaire réalisable cyclique compact est équivalent à déterminer l'existence d'une T-coloration d'arêtes cyclique compacte du même graphe G.

Soit une k-coloration d'arêtes c du graphe G et un sommet v. La *déficience de c en v*, notée $d_v(G, c)$, est le nombre minimum d'entiers à rajouter à $\{c(e) : e \in E_v\}$ pour en faire une ensemble d'entiers consécutifs. La *déficience de c* est définie comme la somme $d(G, c) = \sum_{v \in V} d_v(G, c)$. D'où, c est (linéaire) compact si et seulement si $d(G, c) = 0$. La *déficience* d'un graphe G, notée $d(G)$, est la déficience minimale $d(G, c)$ sur l'ensemble des colorations d'arêtes c de G. Ce concept définit une mesure qui indique à quel point G est proche d'être colorable de manière compacte. En effet, $d(G)$ est le nombre minimum d'arêtes pendantes à rajouter à G pour que le graphe résultant soit colorable compact. Finalement, $S(G)$ est le plus grand entier tel que G admette une $S(G)$-coloration d'arêtes c de déficience $d(G, c) = d(G)$ et utilisant toutes les $S(G)$ couleurs.

Dans l'exemple précédent de *open shop* dans le cas linéaire, si les temps d'attente des jobs et des processeurs ne sont pas interdits, leur nombre devant cependant être minimisé, alors le

problème devient équivalent à trouver une coloration d'arêtes de G avec déficience minimum.

Pour terminer, un *automorphisme* d'un graphe $G = (V, E)$ est une permutation σ de l'ensemble de ses sommets V tel que $(u, v) \in E \Leftrightarrow (\sigma(u), \sigma(v)) \in E$. Ce nouvel ordre des sommets de G préserve donc la matrice d'adjacence.

1.2 Éléments de la problématique

Le problème de déterminer si un graphe admet une k-coloration d'arêtes linéaire compacte (k-LCCP) est fréquent dans les problèmes de confection d'horaire avec contraintes de compacité (Giaro et al., 1999a). Il est $\mathcal{NP}$-complet (Sevastianov, 1990), même pour les graphes bipartis.

Si $k' > k$ et que G admet une k-coloration d'arêtes linéaire compacte, alors il admet aussi un k'-coloration d'arêtes linéaire compacte. Par contre, cette propriété n'est pas vraie en général pour le cas cyclique compact, comme avec le triangle et le pentagone qui admettent une 3-coloration d'arêtes cyclique compacte, mais pas une 4-coloration d'arêtes cyclique compacte.

De plus, une k-coloration d'arêtes linéaire compacte est également cyclique compacte (avec $a_v = c_{min}(v)$ et $b_v = c_{max}(v)$), mais l'inverse n'est pas nécessairement vrai. Par exemple, la 3-coloration d'arêtes du triangle de la Figure 1.1 est cyclique compacte, mais pas linéaire compacte (car la couleur 1 est absente dans $\{c(e)|e \in E_b\}$). Il n'est pas difficile de remarquer que le triangle n'est pas linéaire compact colorable.

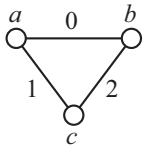


Figure 1.1 Une 3-coloration d'arêtes cyclique compacte qui n'est pas linéaire cyclique.

Le problème de déterminer si un graphe admet une coloration d'arêtes cyclique compacte (CCCP) a été démontré $\mathcal{NP}$-complet (Kubale and Nadolski, 2005), à l'aide d'une réduction en temps polynomial de tout problème de LCCP en un CCCP. Cette réduction ne préserve pas le nombre k de couleurs utilisées. Par contre, elle préserve la bipartition, ce qui démontre que le CCCP est $\mathcal{NP}$-complet même pour le cas biparti.

Le théorème de Vizing (Vizing, 1964) garantit l'existence d'une k-coloration d'arêtes pour tout $k \geq \Delta(G) + 1$ couleurs. Par exemple, il n'est pas difficile de vérifier que la clique K_5 sur cinq sommets n'admet pas de coloration d'arêtes compacte avec $\Delta(K_5) = 4$ couleurs, et

que toute 5-coloration d'arêtes c de K_5 a une déficience de $d(K_5, c) = 3$. Cependant, comme illustré sur la Figure 1.2, il est facile de colorier les arêtes de K_5 avec six couleurs et une déficience de 2.

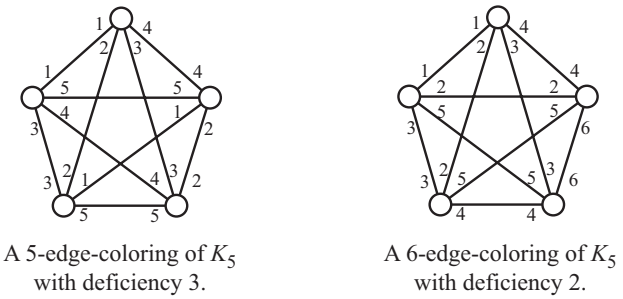


Figure 1.2 La déficience minimum du K_5 ne peut être obtenue qu'en utilisant au moins $\Delta(G) + 2$ couleurs.

Il n'existe pas encore d'algorithme exact efficace pour déterminer la déficience d'un graphe. Le premier défi rencontré pour résoudre ce problème est l'absence de borne supérieure K raisonnable sur le nombre minimum de couleurs à utiliser pour pouvoir colorier le graphe de manière optimale. Certaines modélisations requièrent une telle borne K, lorsqu'une famille de variables identiques est définie pour chaque couleur permise. Un exemple est une modélisation en Programmation en Nombres Entiers, avec une variable binaire $c_{e,k} \in \{0, 1\}$ pour représenter la coloration d'une arête $v \in E$ par la couleur $0 \leq k \leq K$, dans une contrainte du genre $\sum_{0 \leq k \leq K} c_{e,k} = 1$. Un nombre K trop petit donnera une k-coloration déficiente optimale pour ce nombre de couleur K, mais non-optimale en général, comme vu précédemment pour le K_5. Et, dans le cas opposé, un trop grand nombre de couleurs va impliquer la génération d'un nombre important de variables inutiles et leurs contraintes associées, ce qui va naturellement se répercuter sur la performance d'un solveur utilisé pour résoudre l'instance, incluant son utilisation de mémoire. Il est bien sûr possible de modéliser ce problème sans avoir besoin d'une telle borne, ce qui entraîne d'autres avantages et difficultés. Une borne supérieure sur $S(G)$ serait par exemple une option pour K, mais la seule existante n'est valide que pour les cas sans déficience.

Un deuxième défi important, dans le contexte de la déficience, est le nombre très important de solutions optimales équivalentes (i.e. de même déficience optimale) qu'il existe pour un graphe donné G. La structure du problème en permet souvent plusieurs, et celles-ci peuvent être regroupées en classes d'équivalence lorsque l'une peut être obtenue à partir d'une autre par symétrie, sous l'effet d'un automorphisme du graphe. Par exemple, soit le chemin de longueur 4 noté P_4, constitué de trois arêtes successives e_1, e_2 et e_3, avec l'une de ses solutions exprimées

par le triplet $(c(e_1), c(e_2), c(e_3))$. Il existe un automorphisme pour ce graphe, qui échange e_1 et e_3, et laisse e_2 fixe. Ce graphe a quatre solutions optimales : $s_1 = (0, 1, 0)$, $s_2 = (1, 0, 1)$, $s_3 = (0, 1, 2)$ et $s_4 = (2, 1, 0)$. Elles se partitionnent en trois classes d'équivalence $\{s_1\}$, $\{s_2\}$ et $\{s_3, s_4\}$, et le dernier contient deux solutions symétriques.

Dans le cas d'un graphe de petite taille, le solveur arrive souvent à atteindre la solution optimale très rapidement, c'est alors la preuve d'optimalité qui s'avère être problématique. Ces solutions équivalentes donnent souvent l'illusion à l'algorithme d'énumération implicite qu'une branche pourrait donner une meilleure solution que l'optimale, avant d'avoir été étudiée en profondeur, après quoi il se révèle qu'elles ne font que redonner une autre solution optimale.

1.3 Objectifs de recherche

Dans le but de démontrer plus explicitement le lien entre le k-LCCP et le k-CCCP pour un k fixé, nous proposons dans cette thèse une réduction en temps polynomial d'un k-LCCP en un k-CCCP pour tout graphe G et pour tout entier $k \geq 12$. Pour des valeurs plus petites de k, nous montrons qu'un k-LCCP peut être réduit en un 12-CCCP. Notre outil fondamental pour cette réduction est une transformation de graphe qui permet de garantir que deux arêtes non-adjacentes vont être affectées de la même couleur. Comme corollaire, cette transformation nous permet d'imposer ou d'interdire une couleur donnée sur n'importe quelle arête du graphe. Ceci s'avère utile en particulier dans des problèmes d'ordonnancement où certaines tâches doivent être exécutées à des périodes de temps spécifiques. De plus, nous démontrons que déterminer un horaire de production cyclique sans préemption de longueur k (avec possiblement des temps de traitement non-uniformes) est équivalent à résoudre un k-CCCP dans un graphe approprié.

Pour la déficience, nous proposons une borne supérieure pour $S(G)$ valable pour tout graphe, et nous l'utilisons ensuite pour proposer trois modélisations en Programmation en Nombres Entiers pour résoudre ce problème de manière exacte. Nous comparons la performance de ces modélisations entre elles et à une modélisation en Programmation par Contrainte, et déterminons ce dernier comme étant le plus performant et approprié des quatre pour modéliser ce problème. Nous nous concentrons ensuite en particulier sur ce dernier modèle afin de l'améliorer en définissant des méthodes pour y rajouter des contraintes dépendantes du graphe étudié, appelées contraintes GAMBLLE, qui vont permettre de briser une partie de la symétrie présente. Nous analysons leur impact sur la réduction du nombre de solutions optimales et sur le temps de résolution.

1.4 Plan du mémoire

Le présent document est structuré comme suit. Le chapitre 2 regroupe une revue de littérature sur le problème de la déficience et ses variantes, ainsi que des techniques de cassage de symétrie. Le chapitre 3 présente la démarche de l'ensemble du travail de recherche et l'organisation de la thèse. Le chapitre 4 est un article publié dans Discrete Optimization, étudiant la complexité de déterminer la déficience d'un graphe dans un cas de couleurs cycliques. Le chapitre 5 est un article publié dans Computers & Operations Research, comparant des modélisations du problème de la déficience, trois en programmation par nombres entiers et un en programmation par contrainte. Le chapitre 6 est un article soumis à Computers & Operations Research, et perfectionne ce dernier modèle en y ajoutant des contraintes pour briser une partie de la symétrie engendrée par les automorphismes du graphe. Pour terminer, le chapitre 7 présente une discussion plus générale, portant sur l'ensemble du travail accompli, et le chapitre 8 conclut ce document.

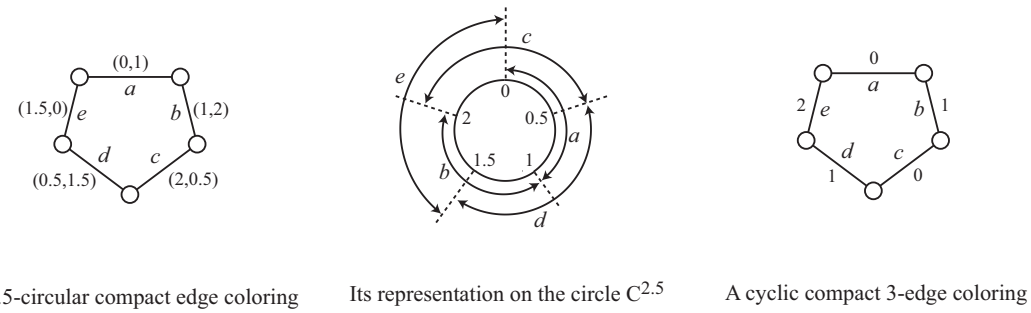
CHAPITRE 2 REVUE DE LITTÉRATURE

Le terme *coloration d'arêtes (linéaire) compacte* est équivalent aux termes *coloration d'arêtes consécutive* (Giaro, 1997; Giaro et al., 2001) et *coloration d'arêtes par intervalle* (Asratian and Casselgren, 2006; Asratian and Kamalian, 1987; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Sevastianov, 1990), utilisé par certains auteurs. Le problème de déterminer si un graphe admet une k-coloration d'arêtes linéaire compacte (k-LCCP) a été proposé par Asratian and Kamalian (1987). C'est une problématique fréquente dans les problèmes de confection d'horaire avec contraintes de compacité (Giaro et al., 1999a). Il est $\mathcal{NP}$-complet (Sevastianov, 1990), même pour les graphes bipartis.

Le problème de déterminer si un graphe admet une k-coloration d'arêtes cyclique compacte (k-CCCP) est étudié dans Nadolski (2008), ainsi que dans Kubale and Nadolski (2005). Dans ce dernier, il est démontré que le CCCP est $\mathcal{NP}$-complet, même pour le cas biparti.

La k-coloration d'arêtes cyclique compacte est à rapprocher de la coloration d'arêtes circulaire compacte définie dans Kubale and Nadolski (2005). Soit un nombre réel positif r, C^r définit le cercle de circonférence r. Soit un point $0 \in C^r$ arbitraire et une orientation, tous les deux arbitraires, (a, b) représente l'arc ouvert commençant en a et se terminant en b, où $a, b \in [0, r]$. La *longueur* d'un arc (a, b) est $b - a$ si $b > a$ et $r - a + b$ si $b < a$. Une coloration d'arêtes r-circulaire est une affectation d'un arc ouvert $A(e) \subset C^r$ de longueur 1 à chaque arête e tel que $A(e)$ et $A(e')$ sont disjoints quand e et e' ont une extrêmité en commun. Une telle coloration est compacte si la fermeture de l'union des arcs ouverts affectées aux arêtes de E_v est un arc sur C^r pour tout sommet v. Quand r est un entier, la coloration d'arêtes r-circulaire compacte est équivalente à la coloration d'arêtes cyclique compacte. Pour les valeurs non-entières de r, la situation est légèrement différente. Par exemple, comme présenté à la Figure 2.1, il existe une coloration d'arêtes r-circulaire compacte du pentagone avec $r = 2.5$, alors qu'il est facile d'observer que toute k-coloration d'arêtes cyclique compacte du pentagone nécessite $k \geq 3$, avec un exemple d'une 3-coloration d'arêtes cyclique compacte présentée à droite dans la Figure 2.1.

Le concept de *déficience* a été proposé par Giaro et al. (1999b), et déterminer la déficience d'un graphe est $\mathcal{NP}$-difficile (Giaro, 1997). Ce problème est aussi étudié dans Asratian and Casselgren (2006); Bouchard et al. (2009); Giaro et al. (2001); Hanson and Loten (1996); Hanson et al. (1998); Pyatkin (2004); Schwartz (2006). À notre connaisance, il n'existe pas encore d'algorithme exact efficace pour déterminer la déficience d'un graphe. Le principal travail dans une direction similaire est une heuristique par Bouchard et al. (2009) basée sur



A 2.5-circular compact edge coloring Its representation on the circle $C^{2.5}$ A cyclic compact 3-edge coloring

Figure 2.1 r-circulaire compact versus cyclique compact pour un non-entier r.

une recherche tabou. Aussi, Giaro et al. (2001) ont prouvé que dans le cas d'un graphe G colorable compactement (i.e. $d(G) = 0$) que $S(G) \leq 2n - 4$, où n est le nombre de sommets de G.

Plusieurs auteurs ont chacun identifié et classifié de manières différentes les nombreux types de symétrie dans leur contexte de recherche respectif. Cette thèse adopte la terminologie de Cohen et al. (2006); Gent et al. (2006), qui classifie en deux catégories générales les symétries de *solution* et de *problème* (ou *contrainte*). De telles permutations de l'ensemble de paires (*variable*,*valeur*) préservent respectivement la solution ou les contraintes du problème. De plus, la dernière est un sous-ensemble de la première. Il est intéressant de relever qu'identifier les symétries du premier type requiert habituellement de trouver d'abord toutes les solutions, alors que celles du deuxième type peuvent être déduites de la structure et de l'expression du problème. Et, par ailleurs, ces deux types ont chacun deux cas spéciaux de symétrie de *variable* et *valeurs*, qui permutent respectivement seulement les variables ou les valeurs.

La *Méthode de Lex-Leader* proposée par Crawford et al. (1996), et par la suite améliorée dans Cohen et al. (2006); Luks and Roy (2004); Pujet (2005), est à la base de la recherche présentée dans ce travail. Elle rajoute des contraintes de façon à ne garder qu'un représentant de chaque classe d'équivalence. Une telle méthode peut requérir que l'on rajoute un énorme nombre de contraintes au modèle, et parfois les rajouter peut s'avérer contre-productif.

CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE ET ORGANISATION DE LA THÈSE

Cette thèse a pour objectif d'étudier et d'améliorer la compréhension d'une famille de problèmes basés sur le concept d'une coloration d'arêtes que l'on essaie de rendre le plus compacte possible. Cette famille demeure peu étudiée, et le chapitre 2 synthétise les travaux précédant le nôtre.

Un des membres de cette famille est la k-coloration d'arête cyclique compacte. Il existait déjà une preuve que ce problème est $\mathcal{NP}$-complet, utilisant une réduction polynomiale à partir du problème de la k-coloration d'arête (linéaire) compacte. Cependant, cette réduction ne conserve pas le nombre de couleurs k. Nous avons donc proposé dans le chapitre 4 une nouvelle réduction polynomiale ayant cette propriété supplémentaire. Notre réduction est fondée sur l'utilisation judicieuse d'un sous-graphe appelé *equalizer*, ayant la propriété de simuler deux arêtes étant forcées à avoir la même couleur dans n'importe quelle coloration d'arête valide du graphe. La portée d'un tel outil dépasse bien largement le cadre de notre réduction, et nous montrons également comment le mettre à profit pour exprimer à l'aide d'une k-coloration d'arête cyclique compacte toutes sortes de variantes dans cette famille de problèmes étudiés.

Un autre volet dans cette famille de problèmes est celui de déterminer la déficience minimale d'une coloration d'arêtes. Nous avons choisi dans le chapitre 5 de tester plusieurs modèles en Programmation en Nombres Entiers et un en Programmation par Contraintes. Ces modèles sont tous destinés à déterminer la déficience de manière exacte, en s'appuyant sur un solveur. Pour rendre possible ces modélisations, nous avons dû déterminer une borne supérieure sur le nombre de couleurs à utiliser. Au terme d'une série de comparaisons de leur performance sur des graphes jusqu'à 100 sommets, il a été déterminé que le modèle en Programmation par Contraintes est clairement le plus performant.

Ces tests ont également mis en lumière une des difficultés majeures pour résoudre ce problème. Dans les graphes de petite taille de neuf sommets ou moins, atteindre une solution optimale est extrêmement aisé. Cependant, prouver l'optimalité de cette solution est ralenti souvent par une myriade de solutions optimales équivalentes. Passer à travers cet ensemble de solutions peu intéressantes concentre l'essentiel du temps de calcul. Une grande majorité de ces solutions équivalentes résultent des automorphismes du graphe examiné. Pour affiner notre procédure précédente, nous avons développé dans le chapitre 6 une méthode pour générer des contraintes aptes à briser une partie de la symétrie dans le modèle. Ces contraintes sont appelées *Graph AutoMorphism Based Lex-Leader Enforcing (GAMBLLE)*, et sont à

ajouter à la formulation en Programmation par Contraintes avant que celle-ci ne soit résolue par un solveur. Ces contraintes s'inspirent de la méthode de Lex-Leader et s'appuient sur une liste complète ou partielle de générateurs pour le groupe d'automorphismes du graphe. Cette démarche donne lieu à un nombre extrêmement réduit de contraintes, obtenues à très peu de frais. Nos tests sur des graphes jusqu'à neuf sommets ont démontré que ce petit nombre améliore le temps de résolution de quelques ordres de magnitude. Cela est dû à la réduction impressionnante de l'espace de solutions, où bon nombre de solutions équivalentes sont éliminées tout en conservant un représentant au minimum de chaque classe d'équivalence.

Pour terminer, le chapitre 7 présente une discussion plus générale, portant sur l'ensemble du travail accompli, et le chapitre 8 conclut ce document.

CHAPITRE 4 ARTICLE 1: ON COMPACT k-EDGE-COLORINGS A POLYNOMIAL TIME REDUCTION FROM LINEAR TO CYCLIC

Abstract

A k-edge-coloring of a graph $G = (V, E)$ is a function c that assigns an integer $c(e)$ (called color) in $\{0, 1, \cdots, k-1\}$ to every edge $e \in E$ so that adjacent edges get different colors. A k-edge-coloring is linear compact if the colors on the edges incident to every vertex are consecutive. The problem $k - LCCP$ is to determine whether a given graph admits a linear compact k-edge coloring. A k-edge-coloring is cyclic compact if for every vertex v there are two positive integers a_v, b_v in $\{0, 1, \cdots, k-1\}$ such that the colors on the edges incident to v are exactly $\{a_v, (a_v + 1)\text{mod } k, \cdots, b_v\}$. The problem $k - CCCP$ is to determine whether a given graph admits a cyclic compact k-edge coloring. We show that the $k - LCCP$ with possibly imposed or forbidden colors on some edges is polynomially reducible to the $k{-}CCCP$ when $k \geq 12$, and to the $12 - CCCP$ when $k < 12$.

4.1 Introduction

All graphs considered in this paper have no loops but may contain parallel edges. A *k-edge-coloring* of a graph $G = (V, E)$ is a function $c : E \to \{0, 1, \cdots, k-1\}$ that assigns a color $c(e)$ to every edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' share a common endpoint. Let E_v denote the set of edges incident with vertex $v \in V$. The *degree* of a vertex v is the number of edges in E_v and the maximum degree in G is denoted $\Delta(G)$. Note that all k-edge-colorings of a graph G use at least $\Delta(G)$ different colors, which means that $\Delta(G) \leq k$.

A k-edge-coloring of a graph $G = (V, E)$ is *linear compact* if $\{c(e) : e \in E_v\}$ is a set of consecutive integers for each vertex $v \in V$. The terms *consecutive edge-colorings* (Giaro, 1997; Giaro et al., 2001) and *interval edge-colorings* (Asratian and Casselgren, 2006; Asratian and Kamalian, 1987; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Sevastianov, 1990) are also used by some authors. A graph is *linearly compactly colorable* if it admits a linear compact k-edge-coloring for some integer k. For a k-edge-coloring c of a graph $G = (V, E)$, 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. It follows from the above definition that if c is linear compact, then $c_{max}(v) = c_{min}(v) + |E_v| - 1$ for all vertices $v \in V$.

A k-edge-coloring is *cyclic compact* if we can associate two positive integers $a_v, b_v < k$ to every vertex v so that $\{c(e) : e \in E_v\} = \{a_v, (a_v + 1)\text{mod } k, \cdots, (a_v + |E_v| - 1)\text{mod } k = b_v\}$

(i.e., color 0 is considered as consecutive to $k-1$). A graph is *cyclically compactly colorable* if it admits a cyclic compact k-edge-coloring for some integer k. While linear compact k-edge-colorings are also cyclic compact (with $a_v = c_{min}(v)$ and $b_v = c_{max}(v)$), the reverse is not necessarily true. For example, the 3-edge-coloring of the triangle shown in Figure 4.1 is cyclic compact but not linear compact (since color 1 is missing in $\{c(e)|e \in E_b\}$). It is not difficult to observe that the triangle is not linearly compactly colorable.

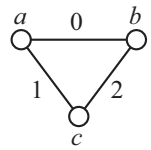
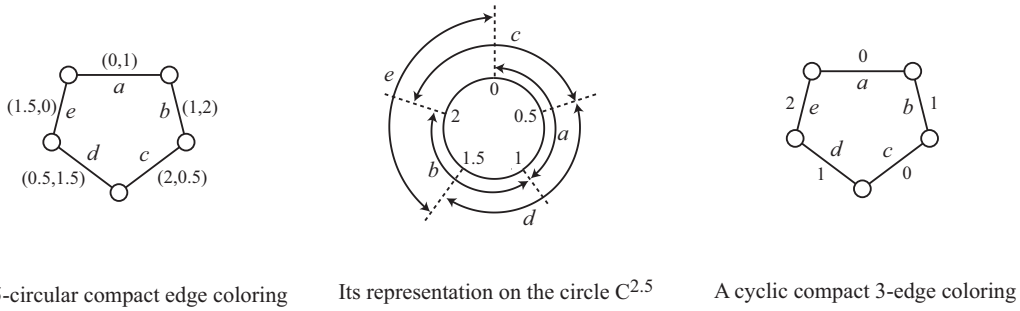


Figure 4.1 A cyclic compact 3-edge-coloring that is not linear compact.

Cyclic compact k-edge-colorings are also studied in (Nadolski, 2008) and are closely related to the circular compact colorability defined in (Kubale and Nadolski, 2005). Given a positive real number r, let C^r denote the circle with circumference r. Taking an arbitrary point $0 \in C^r$ and orientation, we denote (a, b) the open arc starting in a and ending in b, where $a, b \in [0, r]$. The *length* of an arc (a, b) is $b - a$ if $b > a$ and $r - a + b$ if $b < a$. An r-circular edge-coloring is an assignment of an open arc $A(e) \subset C^r$ of length 1 to each edge e so that $A(e)$ and $A(e')$ are disjoint whenever e and e' share a common endpoint. Such a coloring is compact if the closure of the union of open arcs assigned to the edges of E_v is an arc on C^r for each vertex v. When r is integer, compact r-circular edge-colorings are equivalent to cyclic compact r-edge-colorings. For non-integer values of r, the situation is slightly different. For example, as shown in Figure 4.2, there exists a compact r-circular edge-coloring of the pentagon with $r = 2.5$, while it is easy to observe that all cyclic compact k-edge-colorings of the pentagon have $k \geq 3$, an example of a cyclic compact 3-edge-coloring being shown on the right of Figure 4.2.

The problem of determining a linear compact k-edge-coloring (if any) of a graph was introduced by Asratian and Kamalian (1987). It often arises in scheduling problems with compactness constraints (Giaro et al., 1999a). For example, the open shop problem considers m processors $P_1, \cdots, P_m$ and n jobs $J_1, \cdots, J_n$. Each job J_i is a set of s_i tasks. Suppose that each task has to be processed in one time unit on a specific processor. No two tasks of the same job can be processed simultaneously and no processor can work on two tasks at the same time. Moreover, compactness requirements state that waiting periods are forbidden for every job and no idles are allowed on each processor. In other words, the time periods assigned to the tasks of a job must be consecutive, and each processor must be active during



A 2.5-circular compact edge coloring Its representation on the circle $C^{2.5}$ A cyclic compact 3-edge coloring

Figure 4.2 r-circular compact versus cyclic compact for non-integer r.

a set of consecutive periods. The existence of a feasible compact schedule with k time periods is equivalent to the existence of a linear compact k-edge-coloring of the graph G that contains one vertex for each job and each processor, and one edge for each task (i.e., a task of job J_i to be processed on P_j is represented by an edge between the vertices representing J_i and P_j). Each color used in the k-edge-coloring corresponds to a time period. The compactness requirements for each job and each processor are equivalent to imposing that the colors appearing on the edges of E_v must be consecutive for every vertex v in G. In many automated production systems, the production is organized in a cyclic way, i.e., the same production schedule of length T is repeated continuously every T time units. Compactness requirements then impose that the time periods assigned to the tasks of each job and the active period of each processor form a cyclic interval in each production cycle. The existence of a feasible cyclic compact schedule is then equivalent to the existence of a cyclic compact T-edge-coloring of the same graph G.

The problem of determining whether or not a given graph is linearly compactly colorable is denoted $LCCP$ and is known to be $\mathcal{NP}$-complete (Sevastianov, 1990), even for bipartite graphs. Given a k-edge-coloring c of a graph G, let $D_v(G, c)$ denote the minimum number of integers which must be added to $\{c(e) : e \in E_v\}$ to form an interval of consecutive integers. The *deficiency* of c is defined as the sum $D(G, c) = \sum_{v \in V} D_v(G, c)$. Hence, c is linear compact 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 k-edge-colorings c of G (where k can take any positive integer value). This concept, which was introduced by Giaro et al. (1999b), provides a measure of how close G is to be linearly compactly colorable since the deficiency of G is the minimum number of pendant edges that must be added to G such that the resulting graph is linearly compactly colorable. The problem of determining the deficiency of a graph is $\mathcal{NP}$-hard (Giaro, 1997). This problem is also studied in (Asratian and Casselgren, 2006; Giaro et al., 1999b; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Schwartz, 2006; Giaro

et al., 2001; Bouchard et al., 2009).

The problem of determining whether or not a given graph is cyclically compactly colorable is denoted $CCCP$. To demonstrate that it is $\mathcal{NP}$-complete, Kubale and Nadolski (2005) build a graph H from a graph G so that G is linearly compactly colorable if and only if H is cyclically compactly colorable. The graph H is defined as $G \cup K_{1,m+1}$, where m is the number of edges in G and $K_{1,m+1}$ is the star with $m+1$ branches (i.e., the graph containing 1 vertex with degree $m+1$ and $m+1$ vertices with degree 1). In other words, H is obtained from G by adding a new connected component isomorphic to $K_{1,m+1}$. Assume that G is linearly compactly colorable and let k be the smallest integer such that G admits a linear compact k-edge-coloring c. We then have $k \leq m$ since every color in $\{0, \cdots, k-1\}$ appears in c, which means that c can be extended to a cyclic compact $(m+1)$-edge-coloring of H by assigning colors $0, 1, \cdots, m$ to the edges of $K_{1,m+1}$. Also, if H admits a cyclic compact k-edge-coloring c, then $k \geq \Delta(H) = m+1$. Since G contains m edges, at least one of the k colors does not appear in G and, without loss of generality, we may assume that the missing color is $k-1$ (otherwise, a cyclic permutation of the colors in c leads to such a coloring), which means that no vertex v in G has both colors 0 and $k-1$ in E_v. The edge-coloring c restricted to G is therefore linear compact. In summary, G is linearly compactly colorable if and only if H is cyclically compactly colorable, and since H can be obtained from G in polynomial time (by adding $m+2$ vertices and $m+1$ edges), this proves that the $LCCP$ is polynomially reducible to the $CCCP$, which demonstrates the $\mathcal{NP}$-completeness of the $CCCP$. Note that H is bipartite if and only if G is bipartite, which proves that the $CCCP$ is $\mathcal{NP}$-complete even for bipartite graphs.

In this paper, we are interested in determining whether or not a given graph admits a linear (cyclic) compact k-edge-coloring for a fixed integer k.

Definition 1. *Let G be a graph and $k > 0$ an integer.*

- *G is k-linearly compactly colorable if it admits a linear compact k-edge-coloring. We denote k-LCCP the problem of determining whether or not G is k-linearly compactly colorable.*
- *G is k-cyclically compactly colorable if it admits a cyclic compact k-edge-coloring. We denote k-CCCP the problem of determining whether or not G is k-cyclically compactly colorable.*

Note that if $k' > k$ and G is k-linearly compactly colorable, then G is also k'-linearly compactly colorable. However, a k-cyclically compactly colorable graph is not necessarily k'-cyclically compactly colorable for $k' > k$. For example, the triangle and the pentagon are

3-cyclically compactly colorable while they are not 4-cyclically compactly colorable. The reduction proposed in (Kubale and Nadolski, 2005) shows that a given graph G with m edges is m-linearly compactly colorable if and only if H is $(m+1)$-cyclically compactly colorable. We note however that the linear compact m-edge-coloring of G derived from the cyclic compact $(m+1)$-edge-coloring of H uses possibly up to m different colors while less colors may be sufficient. For example, consider the graph G containing a chain on 4 vertices a, b, c, d, with edges between a and b, b and c, and c and d. As shown in Figure 4.3, H is 4-cyclically compactly colorable, and we therefore know that G is 3-linearly compactly colorable. However, the reduction does not provide any information about the 2-linear compact colorability of G which is demonstrated by the linear compact 2-edge-coloring on the right of Figure 4.3.

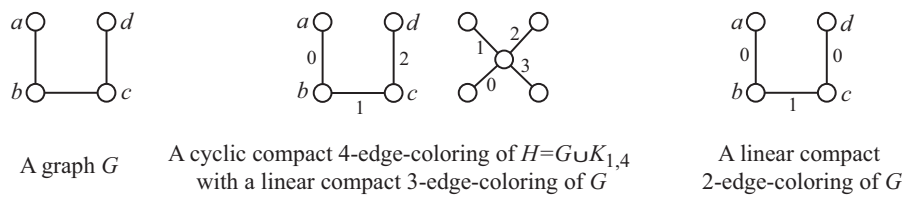


Figure 4.3 Reduction of the $LCCP$ to the $CCCP$.

In this paper, we prove that there is a polynomial time reduction of the $k-LCCP$ to the $k-CCCP$ for every graph G and every integer $k \geq 12$. For smaller values of k, we show that the $k-LCCP$ can be reduced to the $12-CCCP$. These reductions are described in detail in Section 2. A basic tool in our reductions is a graph transformation which makes it possible to impose the same color on two non-adjacent edges. As a corollary, as shown in Section 3, we can impose or forbid a given color on any edge of the graph. This may be helpful when solving production scheduling problems in which some tasks can only be processed at specific time periods. We also show in Section 4 that finding a non-preemptive cyclic production schedule of length k (with possibly non-uniform processing times) is equivalent to solving a $k-CCCP$ in an appropriate graph.

In summary, while the existence of a reduction of the $k-LCCP$ to the $k-CCCP$ is not surprising (since both problems are $\mathcal{NP}$-complete), the proposed reduction, unlike previous transformations, does not increase the number k of colors, and makes it possible to transform a linear coloring problem with forbidden and imposed colors into a cyclic coloring problem without additional constraints.

4.2 Reduction of the $k - LCCP$ to the $k - CCCP$

Note that we always assume $k \geq \Delta(G)$ else G is obviously not k-linearly or k-cyclically compactly colorable. Also, we assume $k \leq |E|$ since it is easy to transform any k-linear or k-cyclic compact coloring of G with $k > |E|$ into a $|E|$-linear or $|E|$-cyclic compact coloring of G. For $k \geq 12$, we describe in this section a construction of a graph H from a graph G so that G is k-linearly compactly colorable if and only if H is k-cyclically compactly colorable. For $k < 12$, we slightly modify our construction of H so that G is k-linearly compactly colorable if and only if H is 12-cyclically compactly colorable.

Given two positive integers $x, y < k$, we denote $[x, y]_k$ the set $\{x \bmod k, (x+1) \bmod k, \cdots, y \bmod k\}$. Hence, a k-edge-coloring c of a graph G is cyclic compact if we can associate two integers $0 \leq a_v, b_v < k$ to every vertex v so that $\{c(e) : e \in E_v\} = [a_v, b_v]_k$.

Figure 4.4 A q-bundle linking vertices u and v.

For an integer $q \geq 1$, we call q-bundle a set of q parallel edges linking two vertices u and v. When drawing a graph, q-bundles are represented as in Figure 4.4. Given a graph G, we build a new graph, denoted G^B by adding a new vertex v' for each vertex v of degree strictly smaller than k in G and by linking each pair v, v' of vertices with a $(k - |E_v|)$-bundle. We denote E_u^B the set of edges incident to a vertex u in G^B. The next lemma shows how to link the $k - LCCP$ in G to the $k - CCCP$ in G^B.

Lemma 1. *G is k-linearly compactly colorable if and only if G^B admits a cyclic compact k-edge-coloring such that for every vertex v of G with $|E_v| < k$, the $(k - |E_v|)$-bundle added to G contains color 0 and/or $k - 1$.*

Proof. ($\Rightarrow$) Let c be a linear compact k-edge-coloring of G and let $[a_v, b_v]_k$ denote the set of colors on the edges of E_v incident to v in G. For vertices v with $|E_v| < k$, let us assign all colors in $[b_v+1, a_v-1]_k$ to the edges of the $(k-|E_v|)$-bundle linking v to v'. The edge-coloring c is thus extended to an edge-coloring c^B of G^B where $\{c^B(e) : e \in E_v^B\} = [0, k-1]_k$ for every v in G, and $\{c^B(e) : e \in E_{v'}^B\} = [b_v + 1, a_v - 1]_k$ for every new vertex v'. Hence, c^B is a cyclic compact k-edge-coloring of G^B. Since c is linear compact, $|[a_v, b_v]_k \cap \{0, k-1\}| \leq 1$ for every vertex v with $|E_v| < k$. Hence, $|[b_v + 1, a_v - 1]_k \cap \{0, k-1\}| \geq 1$ for these vertices, which means that every $(k - |E_v|)$-bundle added to G contains color 0 and/or $k - 1$.

($\Leftarrow$) Let c^B be a cyclic compact k-edge-coloring of G^B such that every $(k-|E_v|)$-bundle added to G contains color 0 and/or $k-1$. For every vertex u in G^B, let $[a_u, b_u]_k = \{c^B(e) : e \in E_u^B\}$. If u is a vertex in G with $|E_u| = k$, then $\{c^B(e) : e \in E_u\} = [0, k-1]_k$. If u is a vertex in G with $|E_u| < k$ then E_u^B contains all edges of E_u as well as those of the $(k-|E_u|)$-bundle linking u to u'. Since $|E_u^B| = k$ and $\{c^B(e) : e \in E_{u'}^B\} = [a_{u'}, b_{u'}]_k$, we have $\{c^B(e) : e \in E_u\} = [b_{u'}+1, a_{u'}-1]_k$. Moreover, $|[a_{u'}, b_{u'}]_k \cap \{0, k-1\}| \geq 1$ implies $|[b_{u'}+1, a_{u'}-1]_k \cap \{0, k-1\}| \leq 1$, which means that the edge-coloring c^B restricted to G is a linear compact k-edge-coloring of G. $\square$

We now have to show how color 0 and/or $k-1$ can be imposed on every $(k-|E_v|)$-bundle added to G. Let t denote the number of vertices v in G with $|E_v| < k$ and let H_t denote the graph with $2t$ vertices $u_1, \cdots, u_t, w_1, \cdots, w_t$ and such that every u_i is linked to w_i $(1 \leq i \leq t)$ with two parallel edges (i.e., a 2-bundle) and every w_i is linked to u_{i+1} $(1 \leq i < t)$ with a $(k-2)$-bundle (see Figure 4.5). Also, let $G^{B+} = G^B \cup H_t$. In other words, G^{B+} is obtained from G^B by adding a new connected component isomorphic to H_t. Let us label $v_1, \cdots, v_t$ the vertices of G with $|E_{v_i}| < k$, let e_i^B be one of the edges of the $(k-|E_{v_i}|)$-bundle linking v_i to v_i', and let e_i^H be one of the two edges linking u_i to w_i. The next lemma provides a link between cyclic compact k-edge-colorings of G^B and G^{B+}.

Figure 4.5 Part of the graph H_t.

Lemma 2. *G^B admits a cyclic compact k-edge-coloring such that every $(k-|E_{v_i}|)$-bundle added to G contains color 0 and/or $k-1$ if and only if G^{B+} admits a cyclic compact k-edge-coloring c with $c(e_i^B) = c(e_i^H)$ for $i = 1, \cdots, t$.*

Proof. ($\Rightarrow$) Let c be a cyclic compact k-edge-coloring of G^B such that every $(k-|E_{v_i}|)$-bundle added to G contains color 0 and/or $k-1$. The colors on every $(k-|E_{v_i}|)$-bundle can be permuted to obtain a new cyclic compact k-edge coloring c' so that e_i^B gets color 0 or $k-1$. Now, assign color $c'(e_i^B)$ to e_i^H, color $k-1-c'(e_i^B)$ (i.e., the other color in $\{0, k-1\}$) to the second edge linking u_i to w_i, and colors $1, \cdots, k-2$ to the $(k-2)$-bundle linking w_i to u_{i+1} (if $i < t$). By construction, the extension of c' to G^{B+} is a cyclic compact k-edge-coloring with $c'(e_i^B) = c'(e_i^H)$ for $i = 1, \cdots, t$.

($\Leftarrow$) Assume now that c is a cyclic compact k-edge-coloring of G^{B+} with $c(e_i^B) = c(e_i^H)$ for $i = 1, \cdots, t$. Since u_1 has degree 2, we can permute the colors in c in a cyclic way to obtain a new cyclic compact k-edge coloring c' so that the two edges incident to u_1 get colors 0 and $k-1$. The $(k-2)$-bundle linking w_1 to u_2 then necessarily contains all colors in $\{1, \cdots, k-2\}$, which means that the 2 edges linking u_2 to w_2 also have color 0 and $k-1$. By repeating the same reasoning, we conclude that the two edges linking u_i to w_i $(1 \leq i \leq t)$ have color 0 and $k-1$ and the edges of the $(k-2)$-bundle linking w_i to u_{i+1} $(1 \leq i < t)$ have colors $1, \cdots, k-2$. Hence, $c'(e_i^H) = 0$ or $k-1$ for $i = 1, \cdots, t$. Since $c'(e_i^B) = c'(e_i^H)$, we conclude that the restriction of c' to G^B is a cyclic compact k-edge-coloring such that each $(k - |E_{v_i}|)$-bundle added to G contains color 0 and/or $k-1$. $\square$

It remains to show how to impose the same color on e_i^B and e_i^H $(1 \leq i \leq t)$. More generally, given two non-adjacent edges e and e' in a graph G, we are interested in building a new graph G' so that G' is k-cyclically compactly colorable if and only if G admits a cyclic compact k-edge-coloring c with $c(e) = c(e')$. For illustration, Figure 4.6 contains two cyclic compact 4-edge-colorings of the same graph. In the left graph, the two edges e and e' have different colors while $c(e) = c(e') = 2$ in the right graph. We are interested in imposing the same color on e and e', which means that the first 4-edge-coloring would not be acceptable.

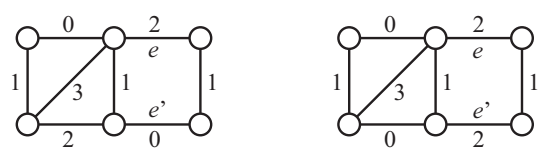


Figure 4.6 Two cyclic compact 4-edge-colorings of a graph.

The next lemma illustrates how q-bundles can be used to impose restrictions on two edges.

Lemma 3. *Let G be a graph and q an integer with $1 \leq q \leq k-3$. Consider three vertices u, v, w in G such that there exist exactly one edge e_{uv} linking u to v, exactly one edge e_{uw} linking u to w, and exactly q edges (i.e. a q-bundle) linking v to w. Assume moreover that v and w are not incident to any other edge in G while u is possibly adjacent to other vertices in G. Let c be a cyclic compact k-edge-coloring of G. Then $c(e_{uw}) = (c(e_{uv}) \pm (q+1)) mod\ k$.*

Proof. Since c is a cyclic compact k-edge-coloring of G, there are four integers a_v, b_v, a_w, b_w such that $\{c(e) : e \in E_v\} = [a_v, b_v]_k$ and $\{c(e) : e \in E_w\} = [a_w, b_w]_k$. Note that $b_v = (a_v + q)$mod k, $b_w = (a_w + q)$mod k, and $[a_w, b_w]_k$ is a subset of $q+1$ consecutive integers (modulo k) chosen in $[a_v, b_v]_k \cup \{c(e_{uw})\}$. Since $q \leq k-3$, we have $a_v - 1 \neq b_v + 1$ (modulo k), and since e_{uv} and e_{uw} are adjacent, this implies that $c(e_{uw}) = (a_v - 1)$mod k or $c(e_{uw}) = (b_v + 1)$mod k.

— if $c(e_{uw}) = (a_v - 1)\text{mod } k$ then $c(e_{uv}) = b_v$ and $[a_w, b_w]_k = [a_v - 1, b_v - 1]_k$; we therefore have $c(e_{uw}) = (b_v - q - 1)\text{mod } k = (c(e_{uv}) - (q + 1))\text{mod } k$.
— if $c(e_{uw}) = (b_v + 1)\text{mod } k$ then $c(e_{uv}) = a_v$ and $[a_w, b_w]_k = [a_v + 1, b_v + 1]_k$; we therefore have $c(e_{uw}) = (a_v + q + 1)\text{mod } k = (c(e_{uv}) + (q + 1))\text{mod } k$.

□

We now consider a more complex structure called *s-shift* that also imposes a restriction on two adjacent edges. Given three vertices u, v, w with an edge e_{uv} between u and v and an edge e_{uw} between u and w, the s-shift structure restricts the color of e_{uw} to $c(e_{uv}) \pm s$, but without imposing any restriction on the other edges incident to v and w. We first give a precise definition of an s-shift.

Definition 2. *Let s be an integer so that $2 \leq s \leq k - 2$. Let u, v, w, u', v', w' be six distinct vertices in a graph G such that there is exactly one edge between u and v, one between u and w, one between u' and v' and one between u' and w'. Assume also that there is a $(k - 2)$-bundle between u and u' and an $(s - 1)$-bundle between v' and w'. Suppose finally that no other edge in G is incident to u, u', v' and w' (while there are possibly other edges incident to v and to w, including one between these two vertices). Let e denote the edge linking u to v and e' the edge linking u to w. Such a structure is called an s-shift between e and e'. It is illustrated in Figure 4.7, with also a simplified representation.*

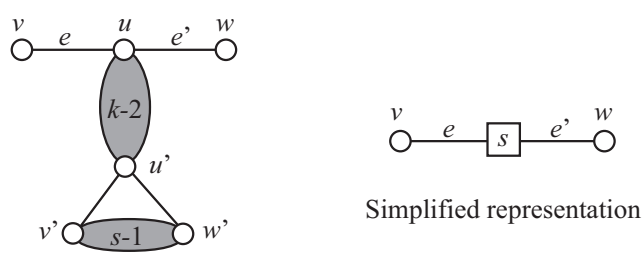


Figure 4.7 An s-shift.

Lemma 4. *Let c be a cyclic compact k-edge-coloring of a graph G and assume there is an s-shift between two edges e and e' in G. Then $c(e') = (c(e) \pm s)\text{mod } k$.*

Proof. Let c be a cyclic compact k-edge-coloring of G and let u, v, w, u', v', w' denote the six vertices of the s-shift between e and e', as in Figure 4.7. By applying Lemma 3 with $q = s - 1$ and the three vertices u', v' and w', and by denoting $e_{u'v'}$ and $e_{u'w'}$ the edges linking u' with v' and u' with w', we get $c(e_{u'w'}) =$

$(c(e_{u'v'}) \pm s)$mod k. It follows that the edges of the $(k-2)$-bundle linking u with u' use all colors in $\{0, \cdots, k-1\} - \{c(e_{u'v'}), c(e_{u'w'})\}$, which means that $\{c(e), c(e')\} = \{c(e_{u'v'}), c(e_{u'w'})\}$. In other words, $c(e') = (c(e) \pm s)$mod k. □

The next structure imposes the same color on four edges. It is called an *equalizer* and is defined as follows.

Definition 3. *Let u, v, u', v' be four distinct vertices linked together according to the structure depicted in Figure 4.8. We assume that all vertices except u, v, u', v' in this structure have no additional edges incident to them. Such a structure is called an equalizer for u, v, u', v'. A simplified representation is also given in Figure 4.8.*

Note that an equalizer contains 28 vertices and $39 + 3k$ edges (since each 6-shift contains 6 vertices and $k + 7$ edges). We now prove that if $k \geq 12$ then the four edges incident to u, v, u' and v' necessarily have the same color in every cyclic compact k-edge-coloring of an equalizer.

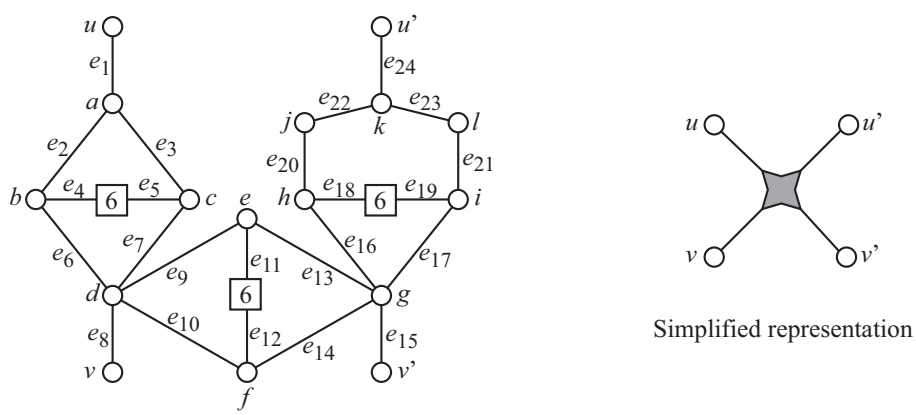


Figure 4.8 An equalizer for u, v, u', v'.

Lemma 5. *Assume $k \geq 12$ and let c be a cyclic compact k-edge-coloring of an equalizer for u, v, u', v'. Then the four edges incident to u, v, u' and v' (i.e, edges e_1, e_8, e_{15}, e_{24} in Figure 4.8) have the same color which can be any integer in $\{0, \cdots, k-1\}$.*

Proof. Let us label the vertices and the edges of the equalizer as in Figure 4.8 and let r be any integer in $\{0, \cdots, k-1\}$. We know from Lemma 4 that $c(e_4) = (c(e_5) \pm 6)$mod k. Without loss of generality, we can assume that $c(e_4) = r - 3$ and $c(e_5) = r + 3$ (all values are taken modulo k).

We first concentrate on vertices u, a, b, c and the 6-shift between b and c. Since b and c are of degree 3, we have $c(e_2) \in [c(e_4) - 2, c(e_4) + 2]_k$ and $c(e_3) \in [c(e_5) - 2, c(e_5) + 2]_k$. In other words, $c(e_2) \in [r - 5, r - 1]_k$ and $c(e_3) \in [r + 1, r + 5]_k$. Since the edges incident to a must have consecutive colors, we necessarily have $c(e_2) = r - 1$, $c(e_1) = r$ and $c(e_3) = r + 1$ if $k > 12$, while the unique other possibility with $k = 12$ is $c(e_2) = r - 5 = r + 7$, $c(e_1) = r + 6$ and $c(e_3) = r + 5$. In both cases, we have $\{c(e_2), c(e_3)\} = \{c(e_1) - 1, c(e_1) + 1\}$ and $\{c(e_4), c(e_5)\} = \{c(e_1) - 3, c(e_1) + 3)\}$.

Consider now the five edges incident to d. If $k > 12$, we have $c(e_6) = r - 2$, $c(e_7) = r + 2$ and $\{c(e_8), c(e_9), c(e_{10})\} = \{r - 1, r, r + 1\}$. If $k = 12$, a second possibility is $c(e_6) = r - 4$, $c(e_7) = r + 4$ and $\{c(e_8), c(e_9), c(e_{10})\} = \{r + 5, r + 6, r + 7\}$. In both cases, the colors on e_8, e_9 and e_{10} are consecutive and equal to those on e_1, e_2, e_3.

We can therefore repeat the same reasoning with the subgraph of the equalizer containing vertices v, d, e, f and the 6-shift between e and f. We get $\{c(e_9), c(e_{10})\} = \{c(e_8) - 1, c(e_8) + 1\}$ and $\{c(e_{11}), c(e_{12})\} = \{c(e_8) - 3, c(e_8) + 3\}$, which means that $c(e_8) = c(e_1)$ and $\{c(e_{11}), c(e_{12})\} = \{r - 3, r + 3\}$. Also, by considering the five edges incident to g, we get the same conclusion as for those incident to d : the colors on e_{15}, e_{16} and e_{17} are consecutive and equal to those on e_8, e_9, e_{10}. By repeating the same reasoning a third and last time on the subgraph of the equalizer containing vertices v', g, h, i and the 6-shift between h and i, we get $\{c(e_{16}), c(e_{17})\} = \{c(e_{15}) - 1, c(e_{15}) + 1\}$, and $\{c(e_{18}), c(e_{19})\} = \{c(e_{15}) - 3, c(e_{15}) + 3)\}$ which means that $c(e_{15}) = c(e_8) = c(e_1)$ and $\{c(e_{18}), c(e_{19})\} = \{r - 3, r + 3\}$.

From $\{c(e_{18}), c(e_{19})\} = \{c(e_{15}) - 3, c(e_{15}) + 3)\}$ and $\{c(e_{16}), c(e_{17})\} = \{c(e_{15}) - 1, c(e_{15}) + 1)\}$, we deduce $\{c(e_{20}), c(e_{21})\} = \{c(e_{15}) - 2, c(e_{15}) + 2)\}$. Since vertex k is of degree 3, we therefore necessarily have $\{c(e_{22}), c(e_{23})\} = \{c(e_{15}) - 1, c(e_{15}) + 1)\}$, which means that $c(e_{24}) = c(e_{15})$. □

Two cyclic compact k-edge-colorings of the equalizer are depicted in Figure 4.9. The first coloring is valid for all $k \geq 12$ while the second coloring demonstrates that the assumption $k \geq 12$ is important. Indeed, a cyclic compact 11-edge-coloring is shown with three different colors on e_1, e_8, e_{15}, e_{24}.

Lemma 6. *Assume $k \geq 12$, let e and e' be two non adjacent edges in a graph G, and let G' be the graph obtained from G by removing e and e' and adding an equalizer for the four endpoints of these two edges. Then G' is k-cyclically compactly colorable if and only if G admits a cyclic compact k-edge-coloring c with $c(e) = c(e')$.*

Proof. Let c' be a cyclic compact k-edge-coloring of G' and let r be the color on the four edges of the equalizer incident to the endpoints of e and e'. By coloring e and e' with color

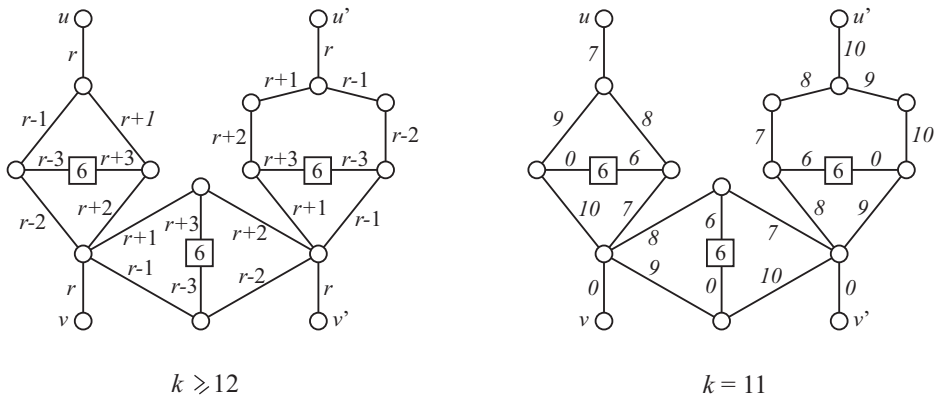


Figure 4.9 Two k-cyclic compact edge-colorings of an equalizer for u, v, u', v'.

r and every other edge of G as in G', one gets a cyclic compact k-edge-coloring c of G with $c(e) = c(e')$.

On the opposite, let c be a cyclic compact k-edge-coloring of G with $c(e) = c(e')$. By coloring the edges of the equalizer as in the left graph of Figure 4.9, with $r = c(e)$, and every other edge of G' as in G, one gets a cyclic compact k-edge-coloring of G'. □

We now complete the description of the polynomial time reduction from the $k - LCCP$ to the $k - CCCP$ for $k \geq 12$. We denote $T(G)$ the graph obtained from G^{B+} by removing the edges e_i^B and e_i^H $(i = 1, \cdots, t)$ and replacing them by an equalizer for v_i, v_i', u_i, w_i . A summary of the transformation is shown in Figure 4.10.

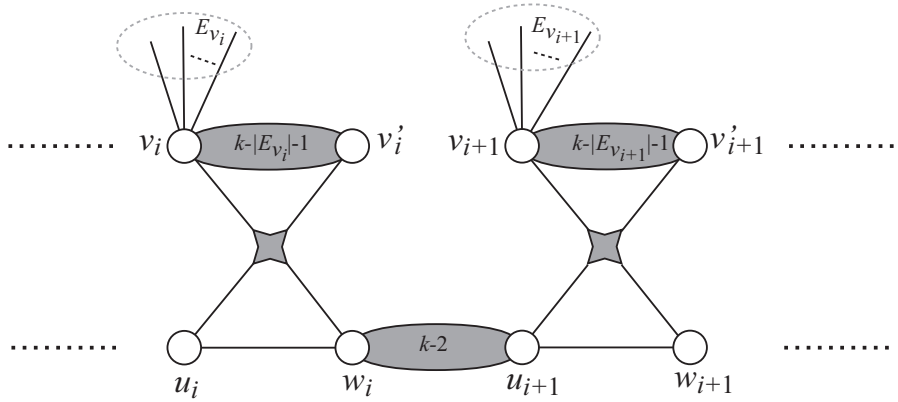


Figure 4.10 Illustration of $T(G)$.

Theorem 1. *Let G be a graph with n vertices and m edges and let $k \geq 12$. Then G is k-linearly compactly colorable if and only if $T(G)$ is k-cyclically compactly colorable. Moreover, the time needed to build $T(G)$ from G is in $O(nk) \subseteq O(nm)$.*

Proof. It follows from Theorems 1 and 2 that G is k-linearly compactly colorable if and only if G^{B+} admits a cyclic compact k-edge-coloring c with $c(e_i^B) = c(e_i^H)$ for $i = 1, \cdots, t$. According to Lemma 6 (applied t times, with $e = e_i^B$ and $e' = e_i^H$ $(i = 1, \cdots, t)$), such a coloring exists if and only if $T(G)$ is k-cyclically compactly colorable.

The graph G^B contains at most $2n$ vertices and $m + \sum_{i=1}^{n}(k - |E_{v_i}|) = kn - m$ edges, where m is the number of edges in G, and the graph H_t contains $2t$ vertices and $kt - (k-2) = k(t-1) + 2$ edges. Since $t \leq n$ we conclude that G^{B+} contains at most $4n$ vertices and $k(2n-1) - m + 2$ edges. As already mentioned, each equalizer contains 24 vertices in addition to u, u', v, v' and $39 + 3k$ edges. Hence, $T(G)$ contains at most $4n + 24n = 28n$ vertices and $k(2n-1) - m + 2 + n(39 + 3k - 2) = k(5n-1) + 37n - m + 2$ edges. In summary, the number of vertices in $T(G)$ is in $O(n)$ and its number of edges is in $O(nk)$. Since we can assume $k \leq m$, the time needed to build $T(G)$ from G is in $O(nk) \subseteq O(nm)$. □

We now show how to transform the $k-LCCP$ to a $12-CCCP$ for $k < 12$. The construction we use is similar to that of $T(G)$ with small variations. First of all, instead of G^B, we build a graph $\tilde{G}^B$ by adding a new vertex v' for each vertex v in G and by linking each pair v, v' of vertices with a $(12 - |E_v|)$-bundle. We then consider the graph $\tilde{H}_n$ (instead of H_t) with $2n$ vertices $u_1, \cdots, u_n, w_1, \cdots, w_n$ and such that each u_i is linked to w_i $(1 \leq i \leq n)$ with a $(12-k)$-bundle and each w_i is linked to u_{i+1} $(1 \leq i < n)$ with a k-bundle. We then define $\tilde{G}^{B+} = \tilde{G}^B \cup \tilde{H}_n$. Finally, let us label $v_1, \cdots, v_n$ the vertices of G, let $e_{i,1}^{\tilde{B}}, \cdots, e_{i,12-k}^{\tilde{B}}$ denote $12-k$ edges of the $(12-|E_{v_i}|)$-bundle linking v_i to v_i' in $\tilde{G}^B$, and let $e_{i,1}^{\tilde{H}}, \cdots, e_{i,12-k}^{\tilde{H}}$ denote the edges linking u_i to w_i in $\tilde{H}_n$. We construct $\tilde{T}(G)$ from $\tilde{G}^{B+}$ by replacing every pair $(e_{i,j}^{\tilde{B}}, e_{i,j}^{\tilde{H}})$ of edges ($i = 1, \cdots, n$; $j = 1, \cdots, 12-k$) by an equalizer for v_i, v_i', u_i, w_i. The construction of $\tilde{T}(G)$ is illustrated on Figure 4.11 for $k = 10$.

Theorem 2. *Let G be a graph with n vertices and let $k < 12$. Then G is k-linearly compactly colorable if and only if $\tilde{T}(G)$ is 12-cyclically compactly colorable. Moreover, the time needed to build $\tilde{T}(G)$ from G is in $O(n)$.*

Proof. The proof is similar as for $k \geq 12$. We only mention the variations. First of all, following the proof of Theorem 1, we can prove in a similar way that G is k-linearly colorable if and only if $\tilde{G}^B$ admits a cyclic compact 12-edge-coloring such that every $(12-|E_v|)$-bundle added to G contains colors $k, k+1, \cdots, 11$. Then following the proof of Theorem 2, we get

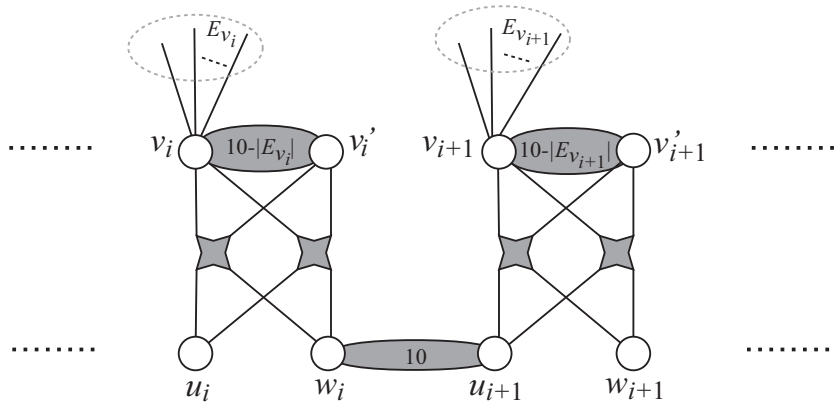


Figure 4.11 Illustration of $\tilde{T}(G)$ for $k = 10$.

that such a coloring exists in $\tilde{G}^B$ if and only if $\tilde{G}^{B+}$ admits a cyclic compact 12-edge-coloring c with $c(e^{\tilde{B}}_{i,j}) = c(e^{\tilde{H}}_{i,j})$ for $i = 1, \cdots, n$ and $j = 1, \cdots, 12-k$. Finally, using Lemma 6, we get that such a coloring exists in $\tilde{G}^{B+}$ if and only if $\tilde{T}(G)$ is 12-cyclically compactly colorable.

The graph $\tilde{G}^B$ contains $2n$ vertices and $m + \sum_{i=1}^{n}(12 - |E_{v_i}|) = 12n - m$ edges, where m is the number of edges in G. Since $\tilde{H}_n$ contains $2n$ vertices and $12n - k$ edges, $\tilde{G}^{B+}$ contains $4n$ vertices and $24n - k - m$ edges. Hence, $\tilde{T}(G)$ contains $4n + 24(12 - k)n$ vertices and $24n - k - m + (12 - k)n(39 + 3k - 2)$ edges, which means that its number of vertices and its number of edges are in $O(n)$. □

4.3 Imposing and forbidding colors

The equalizer described in the previous section is very helpful for imposing or forbidding a color on a given edge. For the open shop problem described in Section 3.1, we can for example impose the period at which a task is performed or we can forbid a subset of periods for some tasks. More details on this application will be given in Section 3.4. We now give more details on how colors can be imposed or forbidden on an edge.

Suppose $k \geq 12$ and assume that every edge e in G has a set H_e of forbidden colors. An empty set H_e for some edge e means that there is no constraint on e. If H_e contains $k-1$ forbidden colors, then the unique color in $\{0, \cdots, k-1\}$ and not in H_e is imposed on e. We define the two following sets F_r and I_r for every $r = 0, \cdots, k-1$:

$$F_r = \{e \in E(G) \mid r \in H_e \text{ and } |H_e| < k-1\}$$
$$I_r = \{e \in E(G) \mid r \notin H_e \text{ and } |H_e| = k-1\}.$$

In words, F_r is the set of edges having r as forbidden color while I_r is the set of edges having r as imposed color. We also consider $F = \bigcup_{r=0}^{k-1} F_r$ and $I = \bigcup_{r=0}^{k-1} I_r$. By definition, $F \cap I = \emptyset$, and the edges in $E - (F \cup I)$ have no restriction. We finally denote $f_r = |F_r|$, $i_r = |I_r|$, and we set $\ell_r = 2(f_r + i_r) + 1$.

We now construct a graph that only depends on the sets F and I. Consider vertices x_j^r for $r = 0, \cdots, k-1$ and $j = 0, \cdots, \ell_r$, as well as two vertices y and z. Link x_{2j}^r to x_{2j+1}^r $(0 \leq j \leq f_r + i_r)$ with a single edge denoted e_j^r and x_{2j-1}^r to x_{2j}^r $(1 \leq j \leq f_r + i_r)$ with a $(k-1)$-bundle. Link also y to z with a k-bundle. Finally, identify every $x_{\ell_r}^r$ with x_0^{r+1}, $r = 0, \cdots, k-2$. The resulting graph is denoted P_H and is illustrated on Figure 4.12. Note that the edge e_0^r exists for $r = 0, \cdots, k-1$.

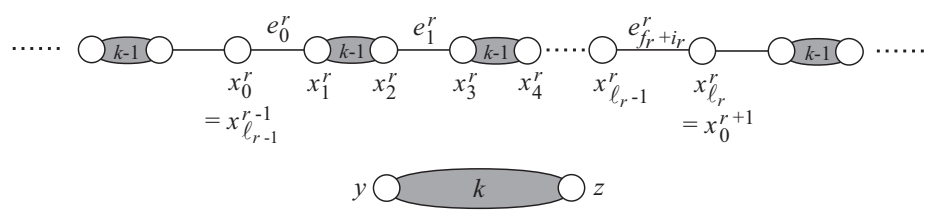


Figure 4.12 Part of the graph P_H.

From P_H we then construct Q_H by replacing k pairs of edges by equalizers. More precisely, for every $r = 0, \cdots, k-1$, we replace one edge of the k-bundle linking y to z and the edge e_0^r linking x_0^r to x_1^r by an equalizer for y, z, x_0^r, x_1^r. Q_H is k-cyclically compactly colorable. Indeed, one can for example assign color r $(0 \leq r < k)$ to every edge e_j^r $(1 \leq j \leq f_r + i_r)$ and all colors in $\{0, \cdots, r-1, r+1, \cdots, k-1\}$ to every $(k-1)$-bundle linking x_{2j-1}^r to x_{2j}^r $(j = 1, \cdots, f_r + i_r)$. Also, every equalizer for y, z, x_0^r, x_1^r can be colored so that the four pendant edges get color r. It is then an easy exercise to check that such a k-edge-coloring is cyclic compact. We now prove that Q_H admits a cyclic compact k-edge-coloring with a special property.

Lemma 7. *Assume $k \geq 12$. For every cyclic compact k-edge-coloring c of Q_H there exist $\alpha \in \{-1, 1\}$ and $\beta \in \{0, \cdots, k-1\}$ such that the k-edge-coloring c' defined by $c'(e) = (\alpha c(e) + \beta) mod\ k$ is cyclic compact with $c'(e_j^r) = r$ for all $r = 0, \cdots, k-1$, $j = 1, \cdots, f_r + i_r$.*

Proof. Let c be a cyclic compact k-edge-coloring of Q_H and let c^P be the cyclic compact k-edge-coloring of P_H obtained from c by coloring e_0^r $(r = 0, \cdots, k-1)$ as the four pendant edges of the equalizer for y, z, x_0^r, x_1^r and by assigning colors $0, \cdots, k-1$ to the edges of the

k-bundle linking y to z. Observe first that $c^P(e_0^r) = c^P(e_j^r)$ for $j = 1, \cdots, f_r + i_r$ since x_{2j-1}^r is linked to x_{2j}^r by a $(k-1)$-bundle. Since $x_{\ell_r}^r = x_0^{r+1}$ is incident to only two edges, we have (modulo k)

$$c^P(e_0^r) = c^P(e_{f_r+i_r}^r) = c^P(e_0^{r+1}) \pm 1 \text{ for all } r = 0, \cdots, k-2. \tag{4.1}$$

Note then that $\{c^P(e_0^0), \cdots, c^P(e_0^{k-1})\} = \{0, \cdots, k-1\}$ since the same set of colors appears on the k-bundle linking y to z. In other words,

$$c^P(e_0^r) \neq c^P(e_0^{r'}) \text{ for all } r \neq r' \tag{4.2}$$

Putting (4.1) and (4.2) together, we get $c^P(e_0^r) = c^P(e_0^0) \pm r$ for all $r = 1, \cdots, k-1$. Consider $\alpha = 1$, $\beta = k - c^P(e_0^0)$ if $c^P(e_0^1) = c^P(e_0^0) + 1$ and $\alpha = -1$, $\beta = c^P(e_0^0)$ if $c^P(e_0^1) = c^P(e_0^0) - 1$. By defining $c'^P(e) = (\alpha c^P(e) + \beta) \text{mod } k$ we get $c'^P(e_j^r) = r$ for $r = 0, \cdots, k-1$, $j = 0, \cdots, f_r + i_r$. Observe that c'^P is cyclic compact since consecutive colors for c^P remain consecutive for c'^P. Hence, by defining $c'(e) = (\alpha c(e) + \beta) \text{mod } k$ for all edges in Q_H we get a cyclic compact k-edge-coloring with $c'(e_j^r) = r$ for all $r = 0, \cdots, k-1$, $j = 1, \cdots, f_r + i_r$. □

For every $r \in \{0, \cdots, k-1\}$ we now use the edges e_j^r $(1 \leq j \leq f_r)$ in Q_H to forbid color r on edges of G and the edges e_j^r $(f_r + 1 \leq j \leq f_r + i_r)$ in Q_H to impose color r on edges of G. This is done as follows. Let $F_r = \{a_1^r, \cdots, a_{f_r}^r\}$ and $I_r = \{b_1^r, \cdots, b_{i_r}^r\}$. For every edge $e \in F$, we consider a graph A_e containing three vertices u_e, v_e, w_e, a single edge linking v_e to u_e and a $(k-1)$-bundle linking v_e to w_e. We then define $R_H(G) = G \cup Q_H \cup \bigcup_{e \in F} A_e$ and we finally transform $R_H(G)$ into a graph $S_H(G)$ as follows:

(i) for every edge e in F, we replace e and the edge linking u_e to v_e by an equalizer for u_e, v_e and the two endpoints of e;

(ii) for every $r = 0, \cdots k-1$ and $j = 1, \cdots, f_r$, we replace e_j^r and one edge linking $v_{a_j^r}$ to $w_{a_j^r}$ by an equalizer for $x_{2j}^r, x_{2j+1}^r, v_{a_j^r}, w_{a_j^r}$;

(iii) for every $r = 0, \cdots, k-1$ and $j = 1, \cdots, i_r$, we replace the two edges b_j^r and $e_{f_r+j}^r$ by an equalizer for $x_{2(f_r+j)}^r, x_{2(f_r+j)+1}^r$ and the two endpoints of b_j^r .

It is not difficult to observe that $S_H(G)$ contains $O(n + mk)$ vertices and $O(mk^2)$ edges, where n is the number of vertices and m the number of edges in G. Part of this construction is illustrated in Figure 4.13. Points (i) and (ii) are used to forbid a color on an edge while point (iii) is used to impose a color. In summary, $S_H(G)$ can be constructed from G in $O(n + mk^2) \subseteq O(n + m^3)$.

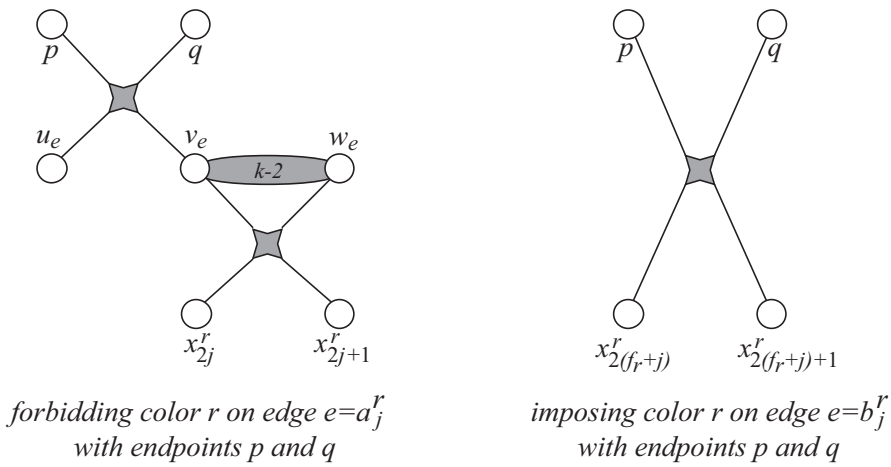


Figure 4.13 Equalizers for forbidding or imposing colors.

Theorem 3. *Assume $k \geq 12$. Then $S_H(G)$ is k-cyclically compactly colorable if and only if G admits a cyclic compact k-edge-coloring c with $c(e) \notin H_e$ for all edges e in G.*

Proof. ($\Leftarrow$) Let c be a cyclic compact k-edge-coloring of G such that $c(e) \notin H_e$ for all edges e in G. We extend c to a k-edge-coloring c^R of R_H as follows:

— for every edge $e \in F$, we color A_e by assigning color $c(e)$ to the edge linking u_e to v_e and the colors in $\{0, \cdots, c(e)-1, c(e)+1, \cdots, k-1\}$ to the $(k-1)$-bundle linking v_e to w_e;

— we color Q_H by assigning color r $(0 \leq r < k)$ to every edge e^r_j $(1 \leq j \leq f_r + i_r)$ and the colors in $\{0, \cdots, r-1, r+1, \cdots, k-1\}$ to every $(k-1)$-bundle linking x^r_{2j-1} to x^r_{2j} $(j = 1, \cdots, f_r + i_r)$. Also, we color every equalizer for y, z, x^r_0, x^r_1 so that the four pendant edges get color r.

We finally define the k-edge-coloring c^S of $S_H(G)$ from c^R by coloring every equalizer so that their pendant edges have the same color as the edges that they replace in R_H. It is easy to check that c^S is a cyclic compact k-edge-coloring of $S_H(G)$.

($\Rightarrow$) Let c^S be a cyclic compact k-edge-coloring of $S_H(G)$ and let c^R be the corresponding cyclic compact k-edge-coloring of R_H obtained by coloring the removed edges $a^r_1, \cdots, a^r_{f_r}$, $b^r_1, \cdots, b^r_{i_r}, e^r_1, \cdots, e^r_{f_r+i_r}$ $(r = 0, \cdots, k-1)$ as the pendant edges of the corresponding equalizers that replace them in $S_H(G)$. Then,

(a) the edge linking u_e to v_e in R_H has color $c^R(e)$ for all $e \in F$;

(b) one edge linking $v_{a^r_j}$ to $w_{a^r_j}$ in R_H has color $c^R(e^r_j)$ (for $r = 0, \cdots, k-1$, $j = 1, \cdots, f_r$);

(c) $c^R(b_j^r) = c^R(e_{f_r+j}^r)$ (for $r = 0, \cdots, k-1$ and $j = 1, \cdots, i_r$).

Consider the restriction c^Q of c^R to Q_H. We know from Lemma 7 that there exist $\alpha \in \{-1, 1\}$ and $\beta \in \{0, \cdots, k-1\}$ such that the k-edge-coloring c'^Q defined by $c'^Q(e) = (\alpha c^Q(e)+\beta)\text{mod } k$ is cyclic compact with $c'^Q(e_j^r) = r$ for all $r = 0, \cdots, k-1$, $j = 1, \cdots, f_r + i_r$.

By performing the same transformation on c^R we get a new cyclic compact k-edge-coloring $c'^R(e) = (\alpha c^R(e) + \beta)\text{mod } k$ of R with the following properties. Let e be any edge in G :

— if $e \in F_r$, then there is an index j $(1 \leq j \leq f_r)$ with $a_j^r = e$. We know from (a) and (b) that $c^R(e) \neq c^R(e_j^r)$, which implies $c'^R(e) \neq r$.
— if $e \in I_r$, then there is an index j $(1 \leq j \leq i_r)$ with $b_j^r = e$. We know from (c) that $c^R(b_j^r) = c^R(e_{f_r+j}^r)$, which implies $c'^R(e) = c'^R(e_{f_r+j}^r) = r$.

In summary, the restriction of c'^R to G is the desired cyclic compact k-edge-coloring of G. □

The same kind of theorem can be stated for k-linear compact edge-colorings of G with imposed and forbidden colors. Indeed, the graph $T(G)$ defined in Section 2 contains two edges incident to u_1: one of these two edges is incident to w_1 while the other one is a pendant edge of the equilazer for v_1, v_1', u_1, w_1. By forbidding colors $\{1, \cdots, k-2\}$ on these two edges, we also forbid that both colors 0 and $k-1$ appear on the edges of a set E_v with $|E_v| < k$ (where E_v denotes the set of edges incident to v in G). Hence, given the sets H_e of forbidden colors on the edges e in G, we define the sets $H'(e)$ of forbidden colors on the edges of $T(G)$ as follows:

— $H'(e) = H(e)$ for all edges e in G;
— $H'(e) = \{1, \cdots, k-2\}$ for the two edges e incident to u_1;
— $H'(e) = \emptyset$ otherwise.

The following corollary is then s a direct consequence of Lemma 2 and Theorems 1 and 3.

Corollary 1. *Assume $k \geq 12$. Then $S_{H'}(T(G))$ is k-cyclically compactly colorable if and only if G admits a linear compact k-edge-coloring c with $c(e) \notin H_e$ for all e in G.*

4.4 Non-preemptive cyclic production scheduling

We now show how the concepts of the previous sections can be used to solve non-premptive cyclic production scheduling problems. Consider the open shop problem with m processors $\mathcal{P}_1, \cdots, \mathcal{P}_m$ and n jobs $J_1, \cdots, J_n$. Each job J_i is a set of s_i tasks. Each task T_{ij} $(j = 1, \cdots, s_i)$ of J_i has to be processed on a given processor $P_{ij} \in \{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$ and has a fixed integer processing time p_{ij}. No two tasks of the same job can be processed simultaneously and no

processor can work on two tasks at the same time. Moreover, compactness requirements state that waiting periods are forbidden for every job and no idles are allowed on each processor. Assume that the production must be organized in a cyclic way, i.e., the same production schedule of length k must be repeated continuously every k time units. It is then imposed that the time periods assigned to each job and the active period of each processor form a cyclic interval in each production cycle. The problem is to determine whether or not such a cyclic schedule exists.

To solve this open shop problem, consider a bipartite graph G with vertex set $\{J_1, \cdots, J_n, \mathcal{P}_1, \cdots, \mathcal{P}_m\}$. For each task T_{ij}, we link J_i to $\mathcal{P}_j$ with p_{ij} parallel edges $e^G_{ij,1}, \cdots, e^G_{ij,p_{ij}}$.

— If preemption is allowed (i.e., the tasks can be interrupted at any integer time), the existence of a feasible cyclic compact schedule of length k is equivalent to the existence of a cyclic compact k-edge-coloring of G.

— If no preemption is allowed, we have to impose consecutive colors on the p_{ij} parallel edges representing each task T_{ij} (color 0 being considered as consecutive to $k-1$). This can be done easily as follows.

  — For each task T_{ij} with $2 \leq p_{ij} \leq k-1$, we add a star S_{ij} to G with p_{ij} branches $e^S_{ij,1}, \cdots, e^S_{ij,p_{ij}}$ to obtain a new graph G^S.

  — We then create $\tilde{G}^S$ from G^S by replacing each pair $(e^G_{ij,r}, e^S_{ij,r})$ of edges ($r = 1, \cdots, p_{ij}$) by an equalizer.

  By Lemma 6, $\tilde{G}^S$ is k-cyclically compactly colorable if and only if G^S admits a cyclic compact k-edge-coloring c with $c(e^G_{ij,r}) = c(e^S_{ij,r})$ for every task T_{ij} with $2 \leq p_{ij} \leq k-1$ and every $r = 1, \cdots, p_{ij}$. Since every center of a star S_{ij} is incident to its p_{ij} branches and to no other edge in G^S, we deduce that the colors $c(e^S_{ij,1}), \cdots, c(e^S_{ij,p_{ij}})$ are consecutive (modulo k), which means that the colors on the p_{ij} parallel edges representing task T_{ij} are consecutive also. We can therefore state the following property.

Property 1. *Assuming $k \geq 12$, there exists a non-preemptive cyclic production schedule of length k if and only if $\tilde{G}^S$ is k-cyclically compactly colorable.*

4.5 Conclusion

We have proved that for $k \geq 12$, the problem of finding a k-linear compact edge-coloring of a graph G with forbidden or imposed colors on some edges is polynomially reducible to the problem of finding a k-cyclic compact edge coloring of another graph. For $k < 12$, we have proposed a reduction to the $12 - CCCP$. As a consequence, production scheduling problems with compactness requirements and in which some tasks can only be processed

at specific time periods may be modeled as a $k - CCCP$. We have also shown how the problem of finding a non-preemptive cyclic production schedule of length k can be modeled as a $k - CCCP$.

The restriction $k \geq 12$ comes from the equalizer which cannot impose the same color on the four pendant edges when $k < 12$. It would be interesting to find a different structure that imposes the same color on several edges without assuming $k \geq 12$.

Note also that while testing linear (cyclic) compact colorability of bipartite graphs is $\mathcal{NP}$-complete ((Sevastianov, 1990; Kubale and Nadolski, 2005)), many production scheduling problems are nevertheless modeled as edge-coloring problems in bipartite graphs. The transformations $T(G)$ and $S_H(G)$ proposed in this paper do not preserve the bipartiteness of the original graph G. In order to be able to use the numerous tools developed by many authors for edge-coloring problems in bipartite graphs, it would therefore be interesting to reduce the $k - LCCP$ in G to the $k - CCCP$ in a graph G' so that G' is bipartite whenever G is.

4.6 Acknowledgements

We would like to thank three anonymous referees for their very helpful comments that improved the quality of the paper.

CHAPITRE 5 ARTICLE 2: A COMPARISON OF INTEGER AND CONSTRAINT PROGRAMMING MODELS FOR THE DEFICIENCY PROBLEM

Abstract

An edge-coloring of a graph $G = (V, E)$ is a function c that assigns an integer $c(e)$ (called color) in $\{0, 1, 2, \dots\}$ to every edge $e \in E$ so that adjacent edges are assigned different colors. An edge-coloring is compact if the colors of the edges incident to every vertex form a set of consecutive integers. The deficiency problem is to determine the minimum number of pendant edges that must be added to a graph such that the resulting graph admits a compact edge-coloring. We propose and analyze three integer programming models and one constraint programming model for the deficiency problem.

5.1 Introduction

All graphs considered in this paper are connected, have no loops, but may contain parallel edges. An *edge-coloring* of a graph $G = (V, E)$ is a function $c : E \to \{0, 1, 2, \dots\}$ that assigns a color $c(e)$ to every edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' share a common endvertex. A *k-edge-coloring* is a similar function, but uses only colors in $\{0, 1, \dots, k-1\}$. Let E_v denote the set of edges incident to vertex $v \in V$. The *degree* $\deg(v)$ of a vertex v is the number of edges in E_v and the maximum degree in G is denoted $\Delta(G)$. Note that all k-edge-colorings of a graph G use at least $\Delta(G)$ different colors, which means that $\Delta(G) \leq k$.

An edge-coloring of a graph $G = (V, E)$ is *compact* if $\{c(e) : e \in E_v\}$ is a set of consecutive nonnegative integers for all vertices $v \in V$. The terms *consecutive edge-colorings* (Giaro, 1997; Giaro et al., 2001) and *interval edge-colorings* (Asratian and Casselgren, 2006; Asratian and Kamalian, 1987; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Sevastianov, 1990) are also used by some authors. A graph is *compactly colorable* if it admits a compact edge-coloring. For an edge-coloring c of a graph $G = (V, E)$, let $\underline{c}(v) = \min_{e \in E_v}\{c(e)\}$ and $\overline{c}(v) = \max_{e \in E_v}\{c(e)\}$ denote, respectively, the smallest and the largest color assigned to an edge incident to v. It follows from the above definitions that if c is compact, then $\overline{c}(v) = \underline{c}(v) + \deg(v) - 1$ for all vertices $v \in V$.

The problem of determining a compact k-edge-coloring (if any) of a graph was introduced by Asratian and Kamalian (1987). Determining whether or not a given graph is compactly colorable is known to be an $\mathcal{NP}$-complete problem (Sevastianov, 1990), even for bipartite

graphs. Given a k-edge-coloring c of a graph G and a vertex v, the *deficiency of c at v*, denoted $d_v(G, c)$, is the minimum number of integers that must be added to $\{c(e) : e \in E_v\}$ to form a set of consecutive integers. The *deficiency* of c is then defined as the sum $d(G, c) = \sum_{v \in V} d_v(G, c)$. Hence, c is compact if and only if $d(G, c) = 0$. The *deficiency of a graph G*, denoted $d(G)$, is the minimum deficiency $d(G, c)$ over all edge-colorings c of G. This concept, which was introduced by Giaro et al. (1999b), provides a measure of how close G is to be compactly colorable. Indeed, $d(G)$ is the minimum number of pendant edges that must be added to G such that the resulting graph is compactly colorable. The problem of determining the deficiency of a graph is $\mathcal{NP}$-hard (Giaro, 1997). This problem is also studied in (Asratian and Casselgren, 2006; Bouchard et al., 2009; Giaro et al., 1999b, 2001; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Schwartz, 2006).

Vizing's theorem (Vizing, 1964) guarantees the existence of a k-edge-coloring for all $k \geq \Delta(G) + 1$. But it may happen that $d(G, c) = d(G)$ only if c uses strictly more than $\Delta(G) + 1$ colors. For example, it is not difficult to verify that the clique K_5 on five vertices has no edge-coloring with $\Delta(K_5) = 4$ colors, and that all 5-edge-colorings c of K_5 have a deficiency $d(K_5, c) = 3$. However, as illustrated in Figure 5.1, it is not difficult to color the edges of K_5 with six colors and deficiency of 2.

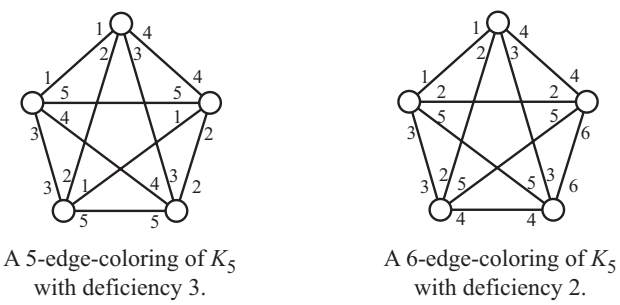


Figure 5.1 The minimum deficiency of K_5 can only be achieved by using at least $\Delta(G) + 2$ colors.

The problem of determining a compact k-coloring of a graph often arises in scheduling problems with compactness constraints (Giaro et al., 1999a). For example, the open shop problem considers m processors $P_1, \ldots, P_m$ and n jobs $J_1, \ldots, J_n$. Each job J_i is a set of s_i tasks. Suppose that each task has to be processed in one time unit on a specific processor. No two tasks of the same job can be processed simultaneously and no processor can work on two tasks at the same time. Moreover, compactness requirements state that waiting periods are forbidden for every job and no idles are allowed on any processor. In other words, the time periods assigned to the tasks of a job must be consecutive, and each processor must be active

during a set of consecutive periods. The existence of a feasible compact schedule with k time periods is equivalent to the existence of a compact k-edge-coloring of the graph G that contains one vertex for each job and each processor, and one edge for each task (i.e., a task of job J_i to be processed on P_j is represented by an edge between the vertices representing J_i and P_j). Each color used in the k-edge-coloring corresponds to a time period. The compactness requirements for each job and each processor are equivalent to imposing that the colors appearing on the edges of E_v must be consecutive for every vertex v in G. If the waiting periods of the jobs and the idles on the processors are not forbidden but their number has to be minimized, the problem is then to find an edge-coloring of G with minimum deficiency.

In this paper, we compare different models for computing the deficiency $d(G)$ of a graph G. In Section 2, we give an upper bound on the number of colors used in an edge-coloring with minimum deficiency. This bound is used to reduce the number of variables in the various models presented in Section 3. The performances of the proposed models are compared in Section 4.

5.2 An upper bound on the number of colors.

Let $s(G)$ be the smallest integer such that G admits an $s(G)$-edge-coloring c with deficiency $d(G, c) = d(G)$. Similarly, let $S(G)$ be the largest integer such that G admits an $S(G)$-edge-coloring c with deficiency $d(G, c) = d(G)$. Note that if G is compactly colorable, then these definitions correspond to those in (Giaro et al., 2001). We clearly have $\Delta(G) \leq s(G) \leq S(G)$. For example, it is not difficult to show that for a chordless cycle C_6 on six vertices, we have $s(G) = 2$ and $S(G) = 4$. k-edge-colorings of C_6 with deficiency $d(C_6) = 0$ are shown in Figure 5.2 for $k = 2, 3, 4$. Note that a graph G does not necessarily admit a k-edge-coloring with deficiency $d(G)$ for all values of $k \in \{s(G), \ldots, S(G)\}$. For example, Sevastianov (1990) has given a graph G with $s(G) = 100$, $S(G) = 173$, and for which there is no k-edge-coloring with minimum deficiency when $k \in \{101, \ldots, 172\}$.

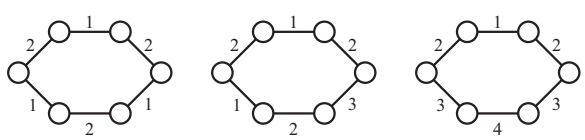


Figure 5.2 k-edge-colorings of C_6 with $k = 2, 3, 4$.

Giaro et al. (2001) have proved that if a graph G is compactly colorable (i.e., $d(G) = 0$), then $S(G) \leq 2n - 4$, where n is the number of vertices in G. We extend this result to all

graphs G, showing that $S(G) \leq 2n - 4 + d(G)$. In their proof, Giaro et al. use the notions of e-paths, i-hairs and i-nodes defined on a graph G' obtained from G by adding two vertices and two edges. We define these notions in a different but equivalent way, since we prefer to work directly with G instead of its augmented graph G'.

For an edge-coloring c, we denote c_{min} and c_{max} the minimum and maximum color used in c. Also, we denote V_c^{min} (respectively V_c^{max}) the subset of vertices incident to an edge having color c_{min} (respectively c_{max}). An e-path P for c is a simple path with vertex set $\{v_1, \ldots, v_p\}$, $p \geq 1$ such that $v_1 \in V_c^{min}$, $v_p \in V_c^{max}$, and every v_i is adjacent to v_{i+1} $(1 \leq i < p)$. We denote by e_i the edge with endvertices v_i and v_{i+1}. Moreover:

- an i-hair $(1 \leq i \leq p)$ of P is an edge e incident to v_i and such that
  - $c(e_{i-1}) < c(e) < c(e_i)$ if $1 < i < p$,
  - $c(e) < c(e_1)$ if $i = 1$,
  - $c(e) > c(e_{p-1})$ if $i = p$;
- an i-node $(1 \leq i \leq p)$ is the endvertex of an i-hair other than v_i.

Let V_P and E_P be the vertex set and the edge set of an e-path P, and let W_P be its set of i-nodes and H_P its set of i-hairs. The skeleton of P is the subgraph G_P of G with vertex set $V_P \cup W_P$ and edge set $E_P \cup H_P$. Clearly, if $d(G, c) = 0$, then the skeleton of every e-path P for c contains at least one edge of color k for each $k \in \{c_{min}, \ldots, c_{max}\}$. Giaro et al. (2001) have proved that if c is an edge-coloring of G with $d(G, c) = d(G) = 0$, then G contains an e-path P for c such that the degree of every i-node in its skeleton G_P is 1 or 2. We are now ready to prove the upper bound on $S(G)$, using a similar proof as in (Giaro et al., 2001).

Theorem 4. *If G is a graph with $n \geq 3$ vertices, then*

$$S(G) \leq 2n - 4 + d(G).$$

Proof. Let c be an edge-coloring of G that uses $S(G)$ colors and such that $d(G, c) = d(G)$. We augment G to $\widetilde{G}$ by introducing $d_v(G, c)$ pendant edges to each deficient vertex v. Hence, a total of $d(G)$ new vertices and $d(G)$ new edges are added to G, and we can now extend c to a compact $S(G)$-edge-coloring $\widetilde{c}$ of $\widetilde{G}$ by assigning the missing colors to the new edges around each vertex v.

Consider an e-path P for $\widetilde{c}$ such that the degree of every i-node in its skeleton $\widetilde{G}_P$ is 1 or 2. Let R be the set of vertices in $\widetilde{G}_P$ that do not belong to G, and let I denote the set of i-nodes of $\widetilde{G}_P$ that do not belong to R. Let p denote the number of vertices in P and let m be the number of edges in $\widetilde{G}_P$. The number of vertices in $\widetilde{G}_P$ is then equal to $p + |I| + |R| \leq n + |R|$.

Note that the i-nodes of I can be of degree 1 or 2 in $\widetilde{G}_P$, but those in R are of degree 1. Since $\widetilde{c}$ uses $S(G)$ colors which all appear on the edges of $\widetilde{G}_P$, we have $S(G) \leq m$, where m is the number of edges in $\widetilde{G}_P$. Hence, it is sufficient to prove that m is not larger than $2n - 4 + d(G)$.

If $p = 1$, then $\widetilde{G}_P$ is a star and $m = |I| + |R| \leq (n-1) + d(G) \leq 2n - 4 + d(G)$. So assume $p \geq 2$. We then have

$$m \leq 2|I| + |R| + p - 1 \leq 2n - (p+1) + |R| \leq 2n - (p+1) + d(G).$$

If $p \geq 3$ then $m \leq 2n - 4 + d(G)$. So assume $p = 2$ while $m \geq 2n - 3 + d(G)$. Then the above inequalities become equalities. Consequently, each i-node in I has degree 2 in $\widetilde{G}_P$ and $n = |I| + 2$. Hence, $\widetilde{G}$ and $\widetilde{G}_P$ have the same vertex set, and $n \geq 3$ implies $|I| > 0$. It follows that the $|I|$ largest possible colors for the 1-hairs that link v_1 to the vertices in I are $\widetilde{c}(e_1) - |I|, \ldots, \widetilde{c}(e_1) - 1$, while the $|I|$ smallest possible colors for the 2-hairs that link v_2 to the vertices in I are $\widetilde{c}(e_1) + 1, \ldots, \widetilde{c}(e_1) + |I|$. Since $\widetilde{c}$ is compact while every vertex in R has degree 1, $\widetilde{G}$ necessarily contains edges that link several pairs of vertices in I. The number of such edges is at least equal to

$$\frac{1}{2}(\sum_{j=1}^{|I|}(c(e_1) + j) - \sum_{j=1}^{|I|}(c(e_1) - j) - \sum_{j=1}^{|I|} 1) = \frac{|I|^2}{2}$$

But the number of such edges cannot be larger than $|I|(|I| - 1)/2$, a contradiction.

□

For illustration, we have $d(K_3) = 1$ and $s(G) = S(G) = 3$. The above bound is therefore the best possible since $2n - 4 + d(G) = 3$ in this case.

5.3 Models

Four different models are described in this section. The first three of them are Integer Linear Programs (IP for short) while the last one is a Constraint Programming model (CP for short). The IP models use Boolean variables $c_{e,k}$ which are equal to 1 if color k is assigned to edge e. In order to reduce the number of variables, we consider an upper bound K on the number of possible colors. By choosing K such that $\Delta(G) \leq K < s(G)$, all edge-colorings produced by our models would have a deficiency strictly larger than $d(G)$. Ideally, we should use a value for K such that $s(G) \leq K \leq S(G)$: by setting $K = s(G)$, we would constrain the problem just enough to ensure the existence of at least one edge-coloring with minimum deficiency,

and with $K = S(G)$ we would not reject any edge-coloring with minimum deficiency. But $s(G)$ and $S(G)$ are typically not known, which explains why we use an upper bound on $S(G)$.

As a corollary of Theorem 4, we have $S(G) \leq 2n-4+D$ for all $D \geq d(G)$. Since, to the best of our knowledge, no graph G with n vertices and $d(G) > n$ is known, we have decided to set $D = n$. In other words, we set $K = 3n - 4$ as an upper bound for $S(G)$. If the minimum deficiency d^* obtained with this upper bound is not larger than n, then we know that the bound is valid, which means that $d(G) = d^*$. Otherwise, if d^* is strictly larger than n, then $n < d(G) \leq d^*$, which means that G is a counter-example to the conjecture that $d(G) \leq n$ for all graphs G with n vertices. In such a case, we can solve the problem a second time, with the new upper bound $K = 2n - 4 + d^*$. The minimum deficiency obtained with this new bound is then equal to $d(G)$.

5.3.1 Integer linear programming models

First IP model (PART)

Model 1 is the most natural one. Let $C = \{0, \ldots, K-1\}$ denote the set of possible colors. As mentioned above, we use Boolean variables $c_{e,k}$ for every pair (e, k), where e is an edge and k is a color in C. We also consider integer variables $\underline{c}_v$, $\overline{c}_v$ and d_v for every vertex v.

The first two constraints define an edge-coloring. The next two constraints impose that $\underline{c}_v$ cannot be larger than the color of an edge incident to v, and that $\overline{c}_v$ cannot be smaller than the color of an edge incident to v. The fifth constraint defines d_v as the number of colors in the interval $[\underline{c}_v, \overline{c}_v]$ that do not appear on edges incident to v. Since we minimize $\sum_{v \in V} d_v$, the values of $\underline{c}_v$ and $\overline{c}_v$ at an optimal solution of Model 1 correspond to the smallest and the largest color assigned to an edge incident to v, while d_v is the deficiency of the coloring at v. The last constraint, although not strictly necessary, forces the usage of color 0 in order to eliminate equivalent solutions obtained by translating the colors in C.

Second IP model (COV)

Model 2 is a variation of the previous one, where we require that *at least* one color is assigned to each edge, instead of *exactly* one. We therefore solve a covering problem rather than a partitioning one.

Model 1 First IP model (PART)

$$
\begin{aligned}
\min \quad & \sum_{v \in V} d_v \\
\text{s.t.} \quad & \sum_{e \in E_v} c_{e,k} \leq 1 && \forall v \in V, k \in C \\
& \sum_{k \in C} c_{e,k} = 1 && \forall e \in E \\
& \underline{c}_v \leq \sum_{k \in C} (k \cdot c_{e,k}) && \forall v \in V, \forall e \in E_v \\
& \overline{c}_v \geq \sum_{k \in C} (k \cdot c_{e,k}) && \forall v \in V, \forall e \in E_v \\
& d_v = \overline{c}_v - \underline{c}_v + 1 - \deg_v && \forall v \in V \\
& \sum_{e \in E} c_{e,0} \geq 1 \\
& c_{e,k} \in \{0, 1\} && \forall e \in E, k \in C \\
& \underline{c}_v \in \{0, \ldots, K - \deg_v\} && \forall v \in V \\
& \overline{c}_v \in \{\deg_v - 1, \ldots, K - 1\} && \forall v \in V \\
& d_v \in \{0, \ldots, K - \deg_v\} && \forall v \in V
\end{aligned}
$$

Model 2 Second IP model (COV)

$$
\begin{aligned}
\min \quad & \sum_{v \in V} d_v \\
\text{s.t.} \quad & \sum_{e \in E_v} c_{e,k} \leq 1 && \forall v \in V, k \in C \\
& \sum_{k \in C} c_{e,k} \geq 1 && \forall e \in E \\
& \underline{c}_v \leq K - (K - k) \cdot c_{e,k} && \forall v \in V, \forall e \in E_v, \forall k \in C \\
& \overline{c}_v \geq k \cdot c_{e,k} && \forall v \in V, \forall e \in E_v, \forall k \in C \\
& d_v = \overline{c}_v - \underline{c}_v + 1 - \deg_v && \forall v \in V \\
& \sum_{e \in E} c_{e,0} \geq 1 \\
& c_{e,k} \in \{0, 1\} && \forall e \in E, k \in C \\
& \underline{c}_v \in \{0, \ldots, K - \deg_v\} && \forall v \in V \\
& \overline{c}_v \in \{\deg_v - 1, \ldots, K - 1\} && \forall v \in V \\
& d_v \in \{0, \ldots, K - \deg_v\} && \forall v \in V
\end{aligned}
$$

Third IP model (STEP)

For Model 3, we replace integer variables $\underline{c}_v$ and $\overline{c}_v$ by Boolean variables $p_{v,k}$ and $q_{v,k}$ defined as follows:

— $p_{v,k} = 0$ if $k \geq \underline{c}_v$, and $p_{v,k-1} \geq p_{v,k}$ for $0 < k \leq \underline{c}_v$;
— $q_{v,k} = 0$ if $k \leq \overline{c}_v$, and $q_{v,k} \leq q_{v,k+1}$ for $\overline{c}_v \leq k < K$.

We therefore have $p_{v,k} + q_{v,k} + \sum_{e \in E_v} c_{e,k} \leq 1$ for all vertices v and all colors $k \in C$. Let $C_v^P = \{0, \ldots, K - 1 - \deg_v\}$ and $C_v^Q = \{\deg_v, \ldots, K - 1\}$. In other words, $p_{v,k}$ ($q_{v,k}$) can have value 1 only if $k \in C_v^P$ (C_v^Q). At a vertex v, we then have $\underline{c}_v \geq \sum_{k \in C_v^P} p_{v,k}$ and $\overline{c}_v \geq K - 1 - \sum_{k \in C_v^Q} q_{v,k}$, which implies

$$\begin{aligned} d_v &= \overline{c}_v - \underline{c}_v + 1 - \deg_v \\ &\geq \left[K - 1 - \sum_{k \in C_v^Q} q_{v,k} \right] - \left[\sum_{k \in C_v^P} p_{v,k} \right] + 1 - \deg_v \\ &= K - \deg_v - \sum_{k \in C_v^P} p_{v,k} - \sum_{k \in C_v^Q} q_{v,k}. \end{aligned}$$

Note that the first, third and fourth constraints of this model impose $p_{v,k} = 0$ if $k \geq \underline{c}_v$ and $q_{v,k} = 0$ if $k \leq \overline{c}_v$. But it could happen that $p_{v,k} = 0$ for $k < \underline{c}_v$ and $q_{v,k} = 0$ for $k > \overline{c}_v$. For example, by setting $p_{v,k} = q_{v,k} = 0$ for all v and all k, we would have a feasible solution. However, minimizing $K - \deg_v - \sum_{k \in C_v^P} p_{v,k} - \sum_{k \in C_v^Q} q_{v,k}$ is equivalent to maximizing $\sum_{k \in C_v^P} p_{v,k} + \sum_{k \in C_v^Q} q_{v,k}$. Hence, at the optimal solution, we necessarily have $p_{v,k} = 1$ for $k < \underline{c}_v$ and $q_{v,k} = 1$ for $k > \overline{c}_v$, which means that

$$\sum_{v \in V} (K - \deg_v - \sum_{k \in C_v^P} p_{v,k} - \sum_{k \in C_v^Q} q_{v,k}) = \sum_{v \in V} d_v.$$

5.3.2 A constraint programming model (CP)

We have implemented a CP model for the deficiency problem. It uses integer variables c_e to denote the color assigned to an edge e. Using *allDifferent* constraints, it is trivial to enforce that edges incident to a vertex must have different colors. Also $\underline{c}_v$ and $\overline{c}_v$ can easily be defined by using the *max* and *min* functions.

Model 3 Third IP model (COV)

$$
\begin{array}{lll}
\min & \sum_{v \in V} (K - \deg_v - \sum_{k \in C_v^P} p_{v,k} - \sum_{k \in C_v^Q} q_{v,k}) & \\
\text{s.t.} & \sum_{e \in E_v} c_{e,k} + p_{v,k} + q_{v,k} \leq 1 & \forall v \in V, k \in C \\
& \sum_{k \in C} c_{e,k} = 1 & \forall e \in E \\
& p_{v,k-1} \geq p_{v,k} & \forall v \in V, k \in C_v^P, k \neq 0 \\
& q_{v,k} \leq q_{v,k+1} & \forall v \in V, k \in C_v^Q, k \neq K-1 \\
& \sum_{e \in E} c_{e,0} \geq 1 & \\
& c_{e,k} \in \{0,1\} & \forall e \in E, k \in C \\
& p_{v,k} \in \{0,1\} & \forall v \in V, k \in C_v^P \\
& p_{v,k} = 0 & \forall v \in V, k \notin C_v^P \\
& q_{v,k} \in \{0,1\} & \forall v \in V, k \in C_v^Q \\
& q_{v,k} = 0 & \forall v \in V, k \notin C_v^Q
\end{array}
$$

Model 4 Constraint Programming (CP)

$$
\begin{array}{lll}
\min & \sum_{v \in V} d_v & \\
\text{s.t.} & \underset{e \in E_v}{\text{allDifferent}} \; c_e & \forall v \in V \\
& \underline{c}_v = \min_{e \in E_v} c_e & \forall v \in V \\
& \overline{c}_v = \max_{e \in E_v} c_e & \forall v \in V \\
& d_v = \overline{c}_v - \underline{c}_v + 1 - \deg_v & \\
& |\{e \in E \mid c_e = 0\}| \geq 1 & \\
& c_e \in C & \forall e \in E \\
& \underline{c}_v \in \{0, \ldots, K - \deg_v\} & \forall v \in V \\
& \overline{c}_v \in \{\deg_v - 1, \ldots, K-1\} & \forall v \in V \\
& d_v \in \{0, \ldots, K - \deg_v\} & \forall v \in V
\end{array}
$$

5.4 Computational results

5.4.1 Experimental setup

In order to compare the performances of the four proposed models, we consider two datasets obtained by using the NAUTY library, originally developed by McKay and Piperno (2013):

Dataset D1. The first dataset is the complete collection of connected simple graphs with $4 \leq n \leq 8$ vertices.

Dataset D2. For a positive integer n and a real number $f \in [0.05, 0.95]$, let G_f^n denote the set of all connected graphs with n vertices and edge density in $[f - 0.05, f + 0.05[$. The second considered dataset is a collection of connected simple graphs, partitioned into subgroups denoted R_f^n. Each subgroup R_f^n contains eight different graphs (when possible) chosen at random in G_f^n. We report results for $n \in \{4, \ldots, 100\}$ and $f \in \{0.1, 0.2, \ldots, 0.9\}$. Note that for $n \leq 7$ and a given f, it may happen that the number of graphs in G_f^n is strictly smaller than 8. In such a case, we set $R_f^n = G_f^n$.

We have run our models respectively with the CPLEX and CP OPTIMIZER solvers (version 12.2) by IBM/ILOG, on a laptop with an Intel Core 2 Duo CPU at 1.20GHz and 4Gb of RAM.

5.4.2 Model comparisons for dataset D1

We first compare the performances of the four models on dataset D1. All models have determined the deficiency of all graphs with at most 7 vertices. For $n = 8$, only the CP model could determine all optimal solutions in a reasonable computing time. An analysis of the optimization process of each model clearly shows that optimal solutions are typically found instantaneously, but a proof of optimality can take minutes or hours in the most difficult cases.

Mean computing times are reported in Table 5.1 for $4 \leq n \leq 7$ vertices. The graphs are grouped according to the value of their deficiency, and the last line of the table indicates the number of graphs in each group. Another representation of the computing times is given in Figure 5.3, where we indicate the number of graphs for which each model had a computing time $t \in [2^i, 2^{i+1}[$, for various values of i. We observe that the CP model is faster than the other models, except for the clique K_7 on 7 vertices, which is the only graph with $n = 7$ vertices and deficiency 3. The PART model comes second.

We reach the same conclusion by comparing the models pairwise in Table 5.2, where we count the number of times a model is faster than another one, with a tolerance of 0.02 seconds. If, for a given graph, the computing time difference between two models is equal to or less than

Table 5.1 Mean computing times for dataset D1, with $n \leq 7$.

n	4	5			6		7			
deficiency	0	0	1	2	0	1	0	1	2	3
PART	0.015	0.033	0.194	0.715	0.112	0.663	0.416	11.267	56.784	330.305
COV	0.019	0.061	0.298	3.051	0.188	2.724	0.803	48.505	462.991	8446.893
STEP	0.034	0.085	0.260	0.449	0.184	1.349	0.938	21.115	122.948	567.281
CP	0.000	0.003	0.004	0.050	0.002	0.011	0.003	0.297	41.842	4169.700
# of graphs	6	15	5	1	104	8	772	75	5	1

Table 5.2 Pairwise comparisons for dataset D1, with $n \leq 7$.

	PART	COV	STEP	CP
PART		830	802	1
COV	121		355	0
STEP	154	538		2
CP	961	988	990	

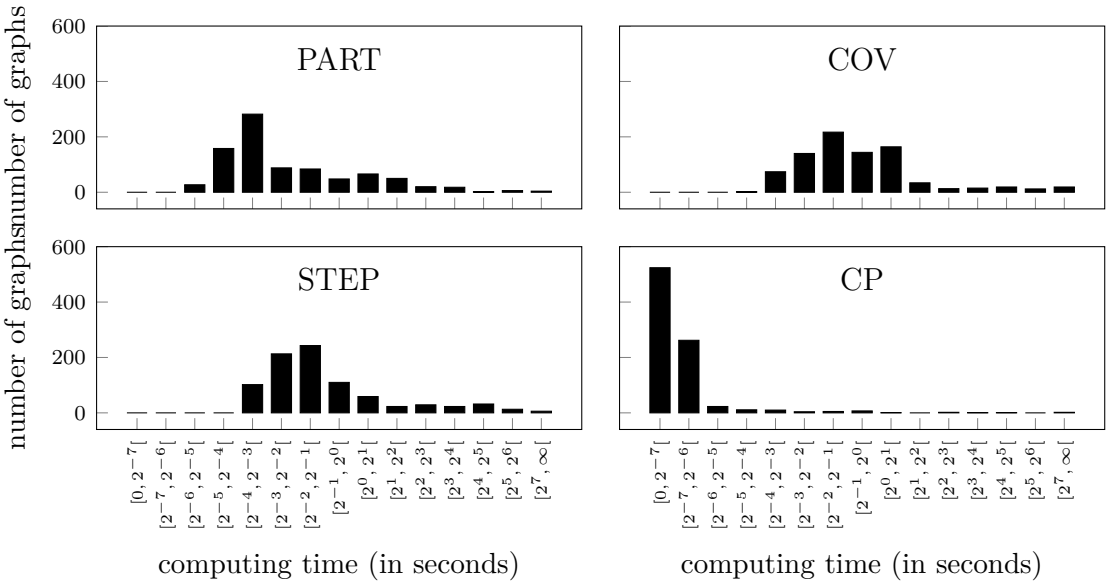


Figure 5.3 Histograms of computing times for dataset D1, with $n \leq 7$.

this tolerance, we consider both models equivalent for this graph. The entry at line A and column B of this table indicates the number of graphs (out of 992) for which the solution time for model A was smaller than that of model B by at least 0.02 seconds.

A more detailed view is given in Figure 5.4, where we indicate the number of graphs for which the difference in computing times lies in an interval $]\frac{2^i}{10}, \frac{2^{i+1}}{10}]$, or its negative equivalent, with a central bin $[-0.02, 0.02]$ for all cases within the tolerance of 0.02 seconds. We clearly see

that the CP model dominates the other ones (almost all bins of the histograms in the last row are on the right side), and that PART is better than COV and STEP (most bins of the upper histograms of the first column are on the left side) . The last two models, seem to have overall comparable performances, even if on some occasions one does better than the other.

As already mentioned above, the CP model is the only one that could determine, in a reasonable computing time, the deficiency of all graphs with $n = 8$ vertices. We show in Table 5.3 the distribution of the graphs with $4 \leq n \leq 8$ vertices, according to their number m of edges and their deficiency. The last line of the table indicates the average computing time needed by the CP model to determine an optimal solution. Also, we show in Figure 5.5 all graphs G with $n = 6$ and 8 vertices, and with largest deficiency. As indicated in Table 5.3 there are eight such graphs for $n = 6$ and four for $n = 8$. Sets of vertices forming either an independent set or a clique, and sharing the same neighborhood outside the set, are grouped in rectangles. An arbitrary permutation of the vertices in a group corresponds to an automorphism of the considered graph, and the existence of these groups therefore indicates that the addition of symmetry breaking constraints in our models could be very helpful for decreasing the computing time.

5.4.3 Model comparisons for dataset D2

For dataset D2, we have decided to compare the four models by limiting the computing time to 10 seconds on each graph. Given one of the models and a set R_f^n we consider the following properties:

— *all solved*: the model could determine the minimum deficiency of all graphs in R_f^n;
— *non solved*: the model has not determined any optimal solution for the graphs in R_f^n;
— *all feasible*: the model could determine a feasible solution for all graphs in R_f^n;
— *non feasible*: the model has not determined any feasible solution for the graphs in R_f^n.

For every $f \in \{0.1, 0.2, \ldots, 0.9\}$, we show in Figure 5.6 the largest n for which the *all solved* and *all feasible* properties were satisfied, and the smallest n for which the *non solved* and *non feasible* properties were satisfied. Note that the *all feasible* and *non feasible* curves for the CP model do not appear on Figure 5.6, the reason being that CP was able to determine feasible solutions for all graphs with 100 vertices. Figure 5.7 superimposes the four *all solved* curves.

The *all solved* curves of Figure 5.7 clearly show that all models struggle with denser graphs. Again, CP does noticeably better than the other models, among which PART seems to have

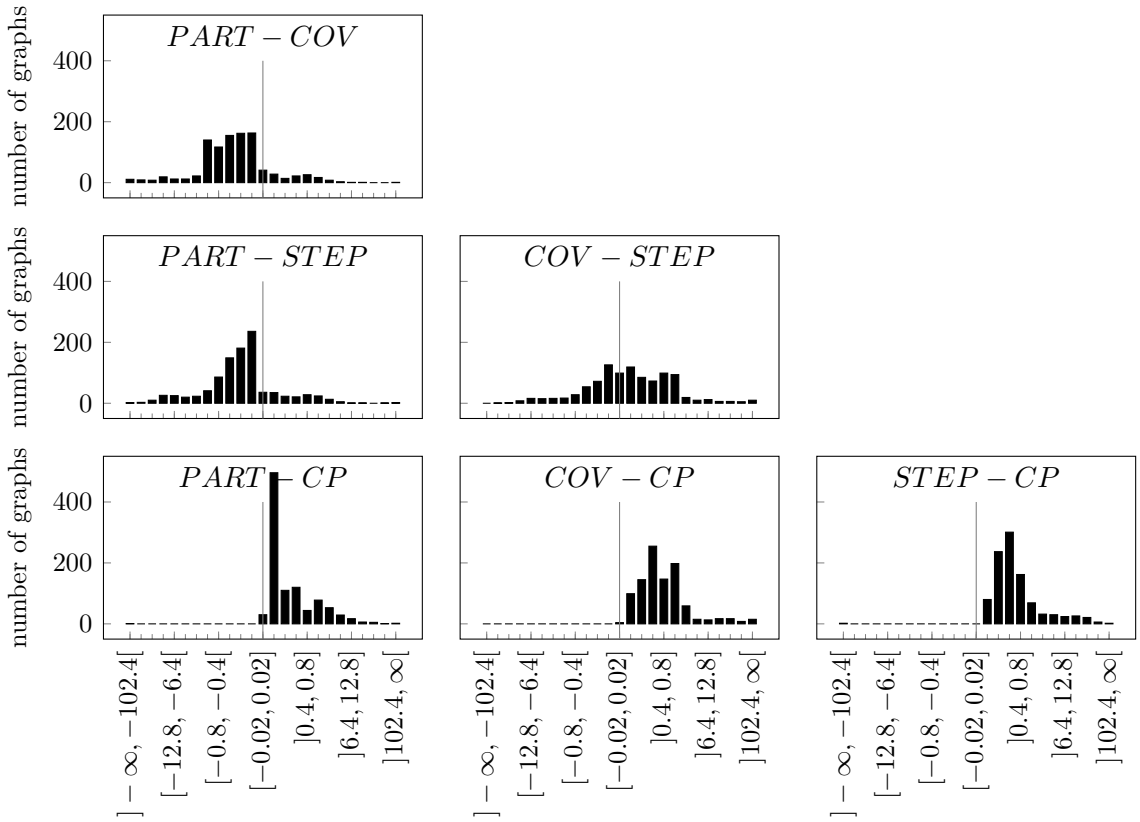


Figure 5.4 Histograms of the differences in computing times for dataset D1, with $n \leq 7$.

a slight advantage. The downward dent at density 0.6 for COV and STEP is due to the existence of a 1-deficient graph in $R^7_{0.6}$ (for which both models need more than 10 seconds to determine the minimum deficiency) while $R^7_{0.5}$ and $R^7_{0.7}$ contain only graphs G with $d(G) = 0$.

When considering the *all feasible* and *non feasible* curves, we can observe that the STEP model performs better than PART and COV. However, the feasible solutions produced by STEP have a much larger deficiency than those produced by the CP model. To illustrate this fact, let $N(f)$ denote the largest number of vertices for which the *all feasible* property is satisfied by STEP for a given f, and let $N'(f) = \lfloor 0.9N(f) \rfloor$. The values of $N'(f)$ are given in Table 5.4. We show in Figure 5.8 the maximum, the minimum and the average deficiency of the feasible edge-colorings produced by STEP and CP for the graphs in R^n_f, with $n = N'(f)$.

Table 5.3 The distribution of all graphs G with $4 \leq n \leq 8$ vertices, according to their number m of edges and their deficiency.

n	4	5			6		7				8		
m deficiency	0	0	1	2	0	1	0	1	2	3	0	1	2
3	2												
4	2	3											
5	1	4	1		6								
6	1	5			13		11						
7		2	2		18	1	32	1			23		
8		1	1		21	1	67				89		
9			1		19	1	101	6			234	2	
10				1	12	2	127	5			483	3	
11					7	2	125	13			797	17	
12					4	1	123	3			1165	4	
13					2		79	16			1412	42	
14					1		54	8	2		1552	27	
15					1		32	8			1483	32	
16							13	8			1274	16	
17							5	4	1		926	44	
18							3	1	1		638	19	1
19								2			376	24	
20									1		215	5	
21										1	103	10	1
22											51	3	2
23											20	4	
24											11		
25											4	1	
26											2		
27											1		
28											1		
# of graphs	6	15	5	1	104	8	772	75	5	1	10860	253	4
mean time	0.000	0.003	0.004	0.050	0.002	0.011	0.003	0.297	41.842	4169.700	0.015	6.865	700.735

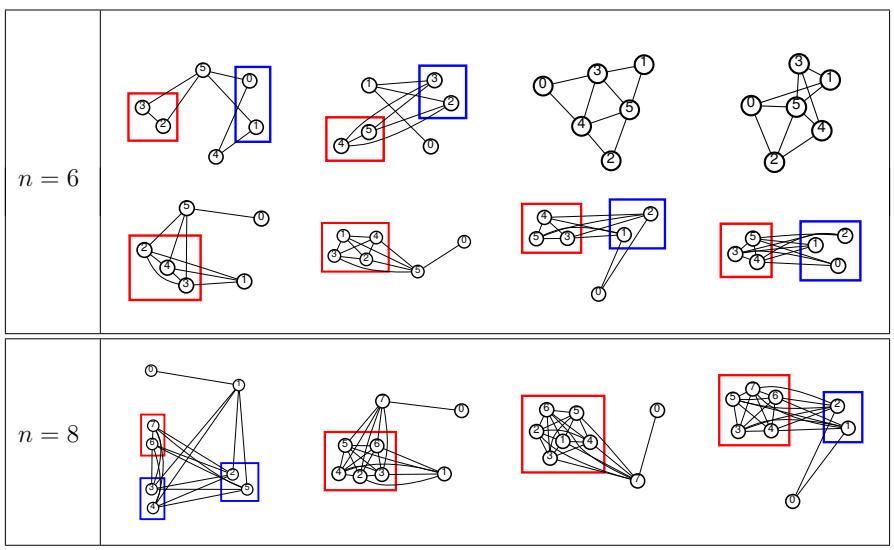


Figure 5.5 All graphs with $n = 6$ and 8 vertices and with largest deficiency.

Table 5.4 Values of $N(f)$ and $N'(f)$.

density	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$N(f)$	79	62	56	51	46	44	39	37	35
$N'(f)$	71	55	50	45	41	39	35	33	31

5.4.4 Other remarks

As mentioned in Section 5.2, the upper bound $K = 3n - 4$ on $S(G)$ is possibly larger than the theoretical bound $2n - 4 + d(G)$. For all graphs G for which we were able to determine the minimum deficiency, we have therefore made a second run of our models, but with $K = 2n - 4 + d(G)$. We could observe a tiny improvement in term of computing time, but nothing meaningful enough that would justify the implementation of a technique that would dynamically decrease the upper bound on $S(G)$ each time a feasible solution is found with deficiency strictly smaller than $K - 2n + 4$.

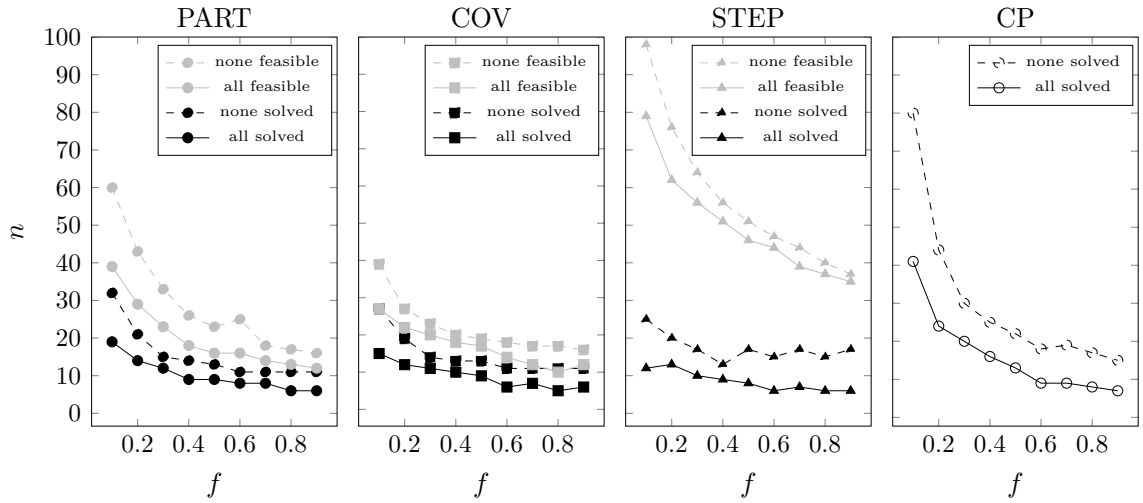


Figure 5.6 The none feasible, all feasible, none solved and all solved curves of the four models.

Also, as already mentioned when analyzing the graphs with $n = 6$ and 8 vertices, and with largest deficiency, symmetry breaking rules can possibly help to decrease the computing time. We have implemented *manually* some of these rules for dense graphs which tend to have many automorphisms and are therefore more challenging for our models. Thanks to the addition of these constraints, we were able to drastically decrease the computing time, which confirms that this is a promising research avenue for the solution of the deficiency problem on larger graphs.

5.5 Conclusion

To limit the number of variables, all models use an upper K on the number of colors to be used. While $K \geq \Delta(G) + 1$ is sufficient to guarantee the existence of an edge-coloring, it may happen that the use of more colors allows a smaller deficiency. To address this problem, we have shown that a coloring with minimum deficiency never uses more than $2n - 4 + d(G)$ colors. This bound on $S(G)$ guarantees that every optimal solution can possibly be obtained by our models. A bound on $s(G)$ would guarantee the existence of at least one edge coloring with minimum deficiency. We have run our models with $K = 3n - 4$, which is a valid bound, unless the optimal value d^* produced by our models is larger than n, in which case we can perform a second run with the valid bound $K = 2n - 4 + d^*$.

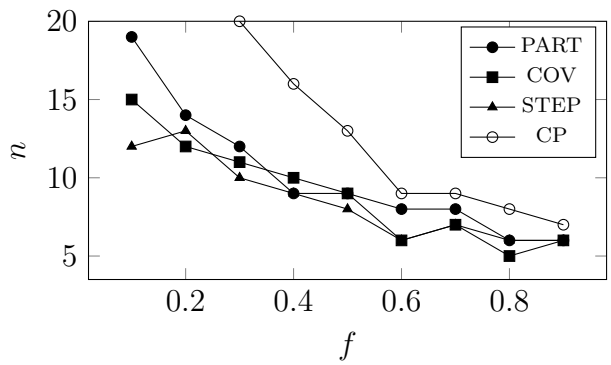


Figure 5.7 The all solved curves.

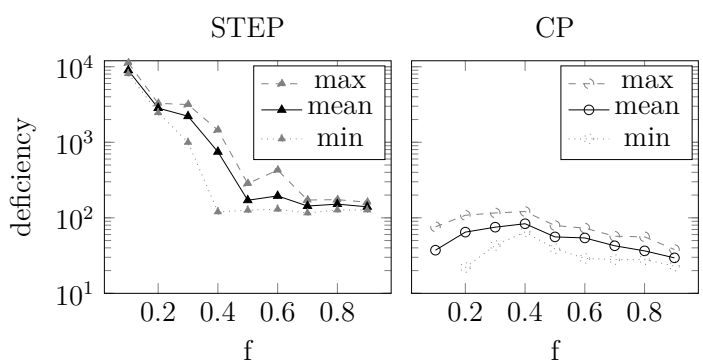


Figure 5.8 Maximum, minimum and average deficiencies of the feasible solutions found by STEP and CP in $R_f^{N'(f)}$.

A comparison of the performances of the four proposed models has clearly shown that the CP model is significantly better than the IP models. We finally found that the addition of symmetry-breaking constraints in our models seems to be a promising research avenue for decreasing the computing time.

CHAPITRE 6 ARTICLE 3: SYMMETRY BREAKING CONSTRAINTS FOR THE MINIMUM DEFICIENCY PROBLEM

Abstract

An edge-coloring of a graph $G = (V, E)$ is a function c that assigns an integer $c(e)$ (called color) in $\{0, 1, 2, \dots\}$ to every edge $e \in E$ so that adjacent edges receive different colors. An edge-coloring is compact if the colors of the edges incident to every vertex form a set of consecutive integers. The minimum deficiency problem is to determine the minimum number of pendant edges that must be added to a graph such that the resulting graph admits a compact edge-coloring.

Because of symmetries, an instance of the minimum deficiency problem can have many equivalent optimal solutions. We present a way to generate a set of symmetry breaking constraints, called GAMBLLE constraints, that can be added to a constraint programming model. The GAMBLLE constraints are inspired by the Lex-Leader ones, based on automorphisms of graphs, and act on families of permutable variables. We analyze their impact on the reduction of the number of optimal solutions as well as on the speed-up of the constraint programming model.

6.1 Introduction

An *edge-coloring* of a graph $G = (V, E)$ is a function $c : E \to \{0, 1, 2, \dots\}$ that assigns a color $c(e)$ to every edge $e \in E$ such that $c(e) \neq c(e')$ whenever e and e' share a common endvertex. Let E_v denote the set of edges incident to vertex $v \in V$. An edge-coloring of $G = (V, E)$ is *compact* if $\{c(e) : e \in E_v\}$ is a set of consecutive nonnegative integers for all vertices $v \in V$.

The problem of determining a compact k-edge-coloring (if any) of a graph was introduced by Asratian and Kamalian (1987). Determining whether or not a given graph admits a compact edge-coloring is known to be an $\mathcal{NP}$-complete problem (Sevastianov, 1990), even for bipartite graphs. Given an edge-coloring c of a graph G and a vertex v, the *deficiency of c at v*, denoted $d_v(G, c)$, is the minimum number of integers that must be added to $\{c(e) : e \in E_v\}$ to form a set of consecutive integers. The *deficiency* of c is then defined as the sum $d(G, c) = \sum_{v \in V} d_v(G, c)$. Hence, c is compact if and only if $d(G, c) = 0$. The *deficiency of a graph G*, denoted $d(G)$, is the minimum deficiency $d(G, c)$ over all edge-colorings c of G. This concept, which was introduced by Giaro et al. (1999b), provides a measure of how close G is to be compactly colorable. Indeed, $d(G)$ is the minimum number of

pendant edges that must be added to G such that the resulting graph is compactly colorable. The *Minimum Deficiency Problem* is to determine $d(G)$. It is an $\mathcal{NP}$-hard problem studied in (Altinakar et al., 2016; Asratian and Casselgren, 2006; Bouchard et al., 2009; Giaro, 1997, 2004; Giaro et al., 1999b, 2001; Hanson and Loten, 1996; Hanson et al., 1998; Pyatkin, 2004; Schwartz, 2006).

As observed in (Altinakar et al., 2016), the problem of determining the deficiency of a small graph is surprisingly hard. The main difficulty is not to generate an optimal solution, but rather to prove its optimality. This is mainly due to the existence of many equivalent optimal solutions. The objective of this paper is to introduce symmetry-breaking constraints, in order to eliminate as many redundant solutions as possible.

Multiple authors have each identified and defined in different ways various types of symmetries in their respective research contexts. This paper adopts the terminology of (Cohen et al., 2006; Gent et al., 2006), which consists of a very general classification into *solution* and *problem* (or *constraint*) symmetries. Such permutations of the set of $(variable, value)$ pairs respectively preserve the solutions or the constraints of the problem. Moreover, the latter is a subset of the former. It is also worth noting that identifying symmetries of the first type usually requires finding all the solutions first, whereas those of the second type can be derived from the structure and expression of the problem. Finally, both of these types allow two special cases, *variable* and *value* symmetry, which only permutes variables or values, respectively.

The *Lex-Leader Method* proposed by Crawford et al. (1996), and later improved in (Cohen et al., 2006; Luks and Roy, 2004; Pujet, 2005), will be at the basis of the research presented here. It adds constraints so as to allow only one member of each equivalence class. Such a method can produce a huge number of constraints and sometimes adding them to a model can be counterproductive. We propose to generate only a subset of these constraints, called GAMBLLE constraints, and analyze their impact on the reduction of the number of optimal solutions.

In (Altinakar et al., 2016), the authors have compared the performances of four models for the minimum deficiency problem. It clearly appears that constrained programming models are significantly better than integer programming ones. The constrained programming model defined in (Altinakar et al., 2016) is described in Section 6.2. Graph automorphisms play an important role in creating symmetries for the minimum deficiency problem. This is illustrated in Section 6.3, and two methods to identify some or all of these automorphisms are proposed in Section 6.4. We then define GAMBLLE constraints in Section 6.5. These are added to the constraint programming model. Computational experiments are reported in

Section 6.6 where we compare eight different ways of adding symmetry breaking constraints for the minimum deficiency problem.

6.2 Model

The symmetries encountered when solving a problem are clearly dependant on the model used to solve it. Following the previous experimentations in (Altinakar et al., 2016), a constraint programming model will be used to solve the minimum deficiency problem. It appears to be much faster than integer programming models and provides a simple correspondence to the graph formulation.

Consider a graph $G = (V, E)$, with vertex set V and edge set E. As mentioned in the introduction, E_v is the set of edges incident to vertex v. We denote by $\deg_v = |E_v|$ the degree of vertex v, and by $\Delta = \max_{v \in V} \deg_v$ the maximum degree of G. Also, C represents a set of colors $\{0, 1, \ldots, K-1\}$ where $K \geq \Delta$, $c_e \in C$ is the color assigned to edge e, and $\underline{c}_v$ and $\overline{c}_v$ respectively denote the minimal and maximal color assigned to an edge incident to vertex v. For the domain of the last two kind of variables, rather than using the entire set C, it is possible to shrink it a little, simply by taking into account the degree of their associated vertex. Finally, d_v is the deficiency at vertex v and $\sum_{v \in V} d_v$ is the deficiency of the coloring. We will use the following constrained programming model proposed in (Altinakar et al., 2016):

$$\min \quad \sum_{v \in V} d_v \tag{6.1}$$

$$\text{s.t.} \quad \underset{e \in E_v}{\text{allDifferent}} \; c_e \qquad \forall v \in V \tag{6.2}$$

$$\underline{c}_v = \min_{e \in E_v} c_e \qquad \forall v \in V \tag{6.3}$$

$$\overline{c}_v = \max_{e \in E_v} c_e \qquad \forall v \in V \tag{6.4}$$

$$d_v = \overline{c}_v - \underline{c}_v + 1 - \deg_v \qquad \forall v \in V \tag{6.5}$$

$$|\{e \in E \mid c_e = 0\}| \geq 1 \tag{6.6}$$

$$c_e \in C \qquad \forall e \in E$$

$$\underline{c}_v \in \{0, \ldots, K - \deg_v\} \qquad \forall v \in V$$

$$\overline{c}_v \in \{\deg_v - 1, \ldots, K-1\} \qquad \forall v \in V$$

$$d_v \in \{0, \ldots, K - \deg_v\} \qquad \forall v \in V$$

Constraints (6.2) to (6.5) are the usual constraints for the minimum deficiency problem

and are sufficient to model it accurately. Constraint (6.6) is added to help breaking the *value* symmetries due to shifting all the colors up or down, by simply enforcing that color 0 should be used at least once. That single extra constraint already immensely improves the performance of the model, and does not interfere with the future ones dealing with *variable* symmetries based on graph automorphisms that is the subject of the rest of this paper.

It was proved in (Altinakar et al., 2016) that if G is a graph with n vertices, then all edge-colorings with minimum deficiency $d(G)$ use at most $2n-4+d(G)$ colors. A fixed number of colors $K = 3n-4$ can therefore be used for all practical purposes, based on the conjecture that the minimum deficiency $d(G)$ of G is always at most equal to n. If the conjecture is proven wrong and the model is applied to a graph with minimum deficiency smaller than n, then the optimal value $D = \sum_{v \in V} d_v$ produced by the model will be larger than n, and we can then run the model a second time, using $K = 2n-4+D$ instead of $3n-4$ to determine the minimum deficiency of the considered graph.

6.3 Graph automorphisms

An *automorphism* of a graph $G = (V, E)$ is a permutation σ of its vertex set V such that $(u, v) \in E \Leftrightarrow (\sigma(u), \sigma(v)) \in E$. This reordering of the vertices of G thus preserves the adjacency matrix. The set of all automorphisms of a graph G forms the permutation group $Aut(G)$. This group acts naturally on V, by its definition. More interestingly, it can act on the edge set E by using the *edge action* h_E that maps a permutation $\sigma \in Aut(G)$ of vertices to a permutation $\sigma_E = h_E(\sigma)$ of the edges, where $\sigma_E((u, v)) = (\sigma(u), \sigma(v))$.

In what follows, we use the standard cycle notations for permutations (Rotman, 2002). It expresses a permutation as a product of cycles corresponding to the orbits of the permutation; since distinct orbits are disjoint, this is referred to as a decomposition into disjoint cycles. Cycles of length 1 are commonly omitted from the cycle notation. The identity permutation, which consists only of 1-cycles will be denoted by Id. Two solutions to the minimum deficiency problem that can be obtained one from the other by a permutation in $Aut(G)$ are called *equivalent solutions.*

Consider for example the chain P_4 on four vertices in Figure 6.1(a). The permutation $(v_1, v_4)(v_2, v_3) \in Aut(P_4)$ translates into permutation (e_1, e_3) of the edge set, and these two permutations both indicate that the two bottom optimal edge-colorings s_3 and s_4 are equivalent. As another example, consider the clique K_3 on three vertices in Figure 6.1(b). The two permutations (v_1, v_2) and (v_2, v_3) are generators for the permutation group $Aut(K_3)$, and these two permutations translate into permutations (e_2, e_3) and (e_1, e_2) of the edge set. In

this case, the permutations in $Aut(K_3)$ indicate that all optimal edge-colorings are equivalent.

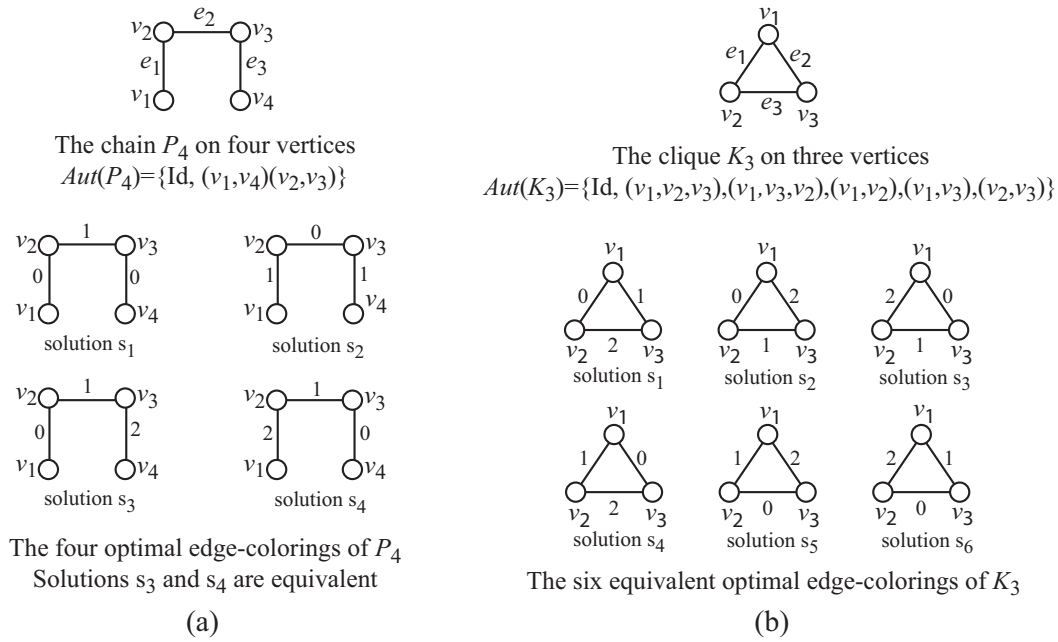


Figure 6.1 Automorphisms and optimal edge-colorings for P_4 and K_3.

$Aut(G)$ also acts on families of vertex (resp. edge) variables which have a one-to-one mapping with the set of vertices (resp. edges) of the graph, such as $\underline{c}_v$, $\overline{c}_v$ and d_v (resp. c_e). For example, consider again the chain on four vertices in Figure 6.1(a). Permutation $(v_1, v_4)(v_2, v_3)$ in $Aut(P_4)$ defines the four following variable permutations : $(\underline{c}_{v_1}, \underline{c}_{v_4})(\underline{c}_{v_2}, \underline{c}_{v_3})$, $(\overline{c}_{v_1}, \overline{c}_{v_4})(\overline{c}_{v_2}, \overline{c}_{v_3})$, $(d_{v_1}, d_{v_4})(d_{v_2}, d_{v_3})$, and (c_{e_1}, c_{e_3}).

6.4 Methods for identifying automorphisms

6.4.1 nauty

Given an input graph G, the NAUTY library created by Brendan McKay(McKay, 1981; McKay and Piperno, 2014) outputs a description of $Aut(G)$ in terms of a generating set. The permutations that are part of this set tend to be fairly simple (when possible, a combination of disjoint transpositions). However, the set is not necessarily minimal for generating $Aut(G)$. For example in Figure 6.2, NAUTY outputs generators $(v_2, v_3), (v_3, v_4), (v_5, v_6), (v_7, v_8)$ and $(v_5, v_7)(v_6, v_8)$; (v_7, v_8) is redundant since it can be obtained by applying $(v_5, v_7)(v_6, v_8)$ followed by (v_5, v_6) and then $(v_5, v_7)(v_6, v_8)$ again.

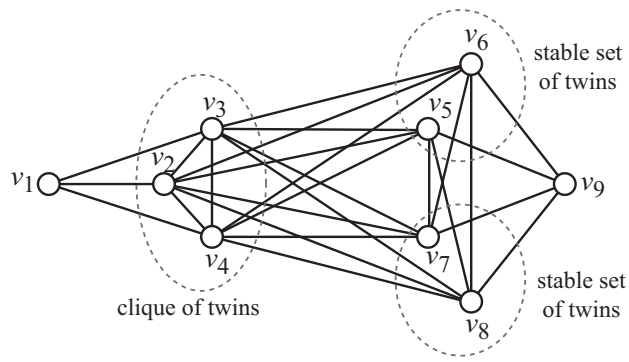


Figure 6.2 A graph with clique and stable sets of twins.

6.4.2 clusters

We define two vertices u and v as *twins* if all vertices $w \neq u, v$ are either adjacent to both u and v or to none of them. A *stable set of twins* is a set of at least two pairwise non-adjacent twins, and a *clique of twins* is a set of at least two pairwise adjacent twins. For illustration, vertices v_2, v_3, v_4 in Figure 6.2 form a clique of twins, while both sets $\{v_5, v_6\}$ and $\{v_7, v_8\}$ are stable sets of twins. The following theorem shows that there is a partition of the vertex set of a graph so that every block of the partition is a maximal stable set of twins, a clique of twins, or a singleton.

Theorem 5. *The intersection of a stable set of twins and a clique of twins is empty.*

Proof. Suppose, by contradiction, that there is a stable set S of twins and a clique K of twins such that $I = S \cap K \neq \emptyset$. Clearly, I contains at most one vertex since vertices in S are non-adjacent, while those in K are adjacent. So let $\{w\} = I$ and let $u \neq w$ be a second vertex in S. All vertices $v \neq w$ in K must be incident to u since u and w have the same set of adacent vertices. But no vertex $v \neq w$ in K can be adjacent to u since v and w have the same neighborhood. Hence K contains only one vertex, a contradiction. □

Finding a partition of the vertex set into maximal stable sets of twins, maximal cliques of twins and singletons is an easy task. Indeed, it is sufficient to observe that every maximal stable set of twins corresponds to a maximal set of identical lines of the adjacency matrix, and every maximal clique of twins corresponds to a maximal set of identical lines of the matrix obtained from the adjacency matrix by changing to 1 every element of its diagonal. All elements that are not in a stable set of twins or in a clique of twins are singletons of the partition. For example, the partition of the vertex set for the graph in Figure 6.2 is $\{\{v_1\}, \{v_2, v_3, v_4\}, \{v_5, v_6\}, \{v_7, v_8\}, \{v_9\}\}$.

Note that a permutation of a subset of vertices in a stable set of twins or in a a clique of twins corresponds to an automorphism. More precisely, let $T = \{v_{i_1}, v_{i_2}, \ldots, v_{i_p}\}$ be a stable set of twins or a clique of twins. The $p-1$ permutations (v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}),..., $(v_{i_{p-1}}, v_{i_p})$ define a set of *generators* for all possible permutations of a subset of vertices in T. We call CLUSTERS the procedure that produces these generators. For the example of Figure 6.2, CLUSTERS would produce the set $\{(v_2, v_3), (v_3, v_4), (v_5, v_6), (v_7, v_8)\}$ of generators. Observe that permutation $(v_5, v_7)(v_6, v_8)$ (i.e. swapping the two stable sets of twins) found by NAUTY cannot be obtained by these generators.

The union of the generators produced by CLUSTERS defines the *cluster-automorphism* group $CAut(G)$, which is a subgroup of $Aut(G)$. Hence, if $\mathcal{C}$ is the set containing all stables sets of twins and all cliques of twins, we have $|CAut(G)| = \prod_{T \in \mathcal{C}}(|T|\,!) \leq |Aut(G)|$, and the total number of generators for $CAut(G)$ is $\sum_{T \in \mathcal{C}}(|T| - 1) \leq n - |\mathcal{C}|$.

6.5 gamblle constraints

The *Lex-Leader constraints*(Crawford et al., 1996) use a vector representation of the variables of a solution, and constrain all permutations of this vector under a set of symmetries to be lexicographically smaller than the first one. This amounts to reducing the solution space to one element for each equivalence class defined by symmetries. Such a method can produce a huge number of constraints, while a subset of these constraints can already be useful, for example when limited to the symmetries due to graph automorphisms.

Consider for example the graph $K_{2,3}$ of figure 6.3 and the following ordering of the variables of the model: $(\underline{c}_{v_1}, \ldots, \underline{c}_{v_5}, \overline{c}_{v_1}, \ldots, \overline{c}_{v_5}, d_{v_1}, \ldots, d_{v_5}, c_{e_1}, \ldots, c_{e_6})$. Both CLUSTERS and NAUTY produce the set $\{(v_1, v_2), (v_3, v_4), (v_4, v_5)\}$ of generators. Permutation (v_3, v_4) imposes the following constraint :

$$\begin{aligned}&(\underline{c}_{v_1}, \underline{c}_{v_2}, \underline{c}_{v_3}, \underline{c}_{v_4}, \underline{c}_{v_5}, \overline{c}_{v_1}, \overline{c}_{v_2}, \overline{c}_{v_3}, \overline{c}_{v_4}, \overline{c}_{v_5}, d_{v_1}, d_{v_2}, d_{v_3}, d_{v_4}, d_{v_5}, c_{e_1}, c_{e_2}, c_{e_3}, c_{e_4}, c_{e_5}, c_{e_6})\\ \preceq_{\text{lex}} &(\underline{c}_{v_1}, \underline{c}_{v_2}, \underline{c}_{v_4}, \underline{c}_{v_3}, \underline{c}_{v_5}, \overline{c}_{v_1}, \overline{c}_{v_2}, \overline{c}_{v_4}, \overline{c}_{v_3}, \overline{c}_{v_5}, d_{v_1}, d_{v_2}, d_{v_4}, d_{v_3}, d_{v_5}, c_{e_2}, c_{e_1}, c_{e_3}, c_{e_5}, c_{e_4}, c_{e_6})\end{aligned}$$

When the head of the solution vector comprises all the elements of a family of permutable variables, part of the effect of a permutation of the vertices or of the edges is a rearrangment of this head. In the above example, the effect of permutation (v_3, v_4) is to change $(\underline{c}_{v_1}, \underline{c}_{v_2}, \underline{c}_{v_3}, \underline{c}_{v_4}, \underline{c}_{v_5}, \ldots)$ into $(\underline{c}_{v_1}, \underline{c}_{v_2}, \underline{c}_{v_4}, \underline{c}_{v_3}, \underline{c}_{v_5}, \ldots)$. The first simplification (called *trimming*) consists of comparing with an inequality only the first pair of different variables, instead of a lexicographical comparison of the whole vectors. In the previous example, the

constraint becomes

$$\underline{c}_{v_3} \leq \underline{c}_{v_4}.$$

If we rearrange the variables so that the head of a solution vector is $(c_{e_1}, c_{e_2}, c_{e_3}, c_{e_4}, c_{e_5}, c_{e_6}, \ldots)$, then permutation (v_3, v_4) changes this head to $(c_{e_2}, c_{e_1}, c_{e_3}, c_{e_5}, c_{e_4} c_{e_6}, \ldots)$ and the constraint resulting from the trimming is

$$c_{e_1} \leq c_{e_2}.$$

In some cases, it is possible to use original constraints in the model to further strengthen a constraint to a strict inequality, as with the edge color variables when the considered edges have an endvertex in common. This results from the fact that such variables appear together in an allDifferent constraint. This special case makes the trimmed constraint equivalent to the full original lexicographical one. In the considered example, the constrained programming model imposes that c_{e_1}, c_{e_2} and c_{e_3} must be all different, and permutation (v_3, v_4) therefore gives

$$c_{e_1} < c_{e_2}.$$

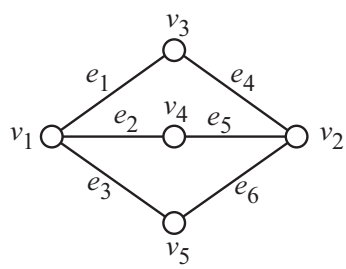


Figure 6.3 The $K_{2,3}$ bipartite graph, with $|CAut(K_{2,3})| = |Aut(K_{2,3})| = 12$.

The second simplification consists in adding constraints for only a few permutations from the automorphism group. Consider a permutation in $Aut(G)$ and let π be its effect on a family of permutable variables. Permutation π can be written as a product of disjoint cycles. Let C be the cycle in π that contains the variable with smallest index, say u_i. For every u_j in C with $j \neq i$ we do the following : if the model does not imply $u_i \neq u_j$, we add inequality $u_i \leq u_j$ to the set of constraints; otherwise, we add the strict inequality $u_i < u_j$. Every permutation π thus produces $|C| - 1$ constraints, to account for the whole orbit of the variable u_i created by repeated application of π. We name the resulting inequalities Graph AutoMorphism-Based Lex-Leader Enforcing (GAMBLLE) constraints. This is summarized in Algorithm 1. Line 3 guarantees that the inequality compares the first differing pair of the underlying lexicographical constraint based on the solution vector headed by the family $\mathcal{F}$ of permutable variables.

Algorithm 1: Generation of GAMBLLE constraints for a family of permutable variables

input : Graph G, a subset $P \subseteq Aut(G)$, a family $\mathcal{F}$ of ordered permutable variables
output : A set L of GAMBLLE inequality constraints

```
Let P^F be the set of permutations of the elements of F obtained from P
foreach π ≠ Id in P^F do
    Let C be the cycle of π with the smallest indexed variable u_i
    foreach u_j ∈ C with j ≠ i do
        if the model implies u_i ≠ u_j then
            Add inequality u_i < u_j to L
        else
            Add the inequality u_i ≤ u_j to L
```

For illustration, consider the graph G in Figure 6.4 with $|Aut(G)| = 3$. It is the smallest graph with all permutations $\pi \neq \mathrm{Id}$ in $Aut(G)$ having no transposition (i.e., cycle with only two elements). NAUTY generates permutation $(v_1, v_4, v_7)(v_2, v_5, v_8)(v_3, v_6, v_9)$ with three cycles. The corresponding permutation of the $\underline{c}_v$ variables is $\pi = (\underline{c}_{v_1}, \underline{c}_{v_4}, \underline{c}_{v_7})(\underline{c}_{v_2}, \underline{c}_{v_5}, \underline{c}_{v_8})(\underline{c}_{v_3}, \underline{c}_{v_6}, \underline{c}_{v_9})$. The cycle with the variable of smallest index is $C = (\underline{c}_{v_1}, \underline{c}_{v_4}, \underline{c}_{v_7})$ and we therefore add constraints

$$\underline{c}_{v_1} \leq \underline{c}_{v_4} \text{ and } \underline{c}_{v_1} \leq \underline{c}_{v_7}.$$

If we are interested in the edge colors, we first have to order the edges. One possible way is to order them using the lexicographical ordering of their pair of endvertices. Hence e_1 is the edge linking v_1 with v_2, e_2 is the edge linking v_1 with v_4, and so on. Such an ordering is shown on the leftside of Figure 6.4. Permutation $(v_1, v_4, v_7)(v_2, v_5, v_8)(v_3, v_6, v_9)$ translates into the following permutation of the c_e variables:

$$(c_{e_1}, c_{e_9}, c_{e_{14}})(c_{e_2}, c_{e_{10}}, c_{e_3})(c_{e_4}, c_{e_7}, c_{e_{12}})(c_{e_5}, c_{e_8}, c_{e_{13}})(c_{e_6}, c_{e_{11}}, c_{e_{15}}).$$

The cycle with smallest index is $C = (c_{e_1}, c_{e_9}, c_{e_{14}})$ and we therefore add constraints

$$c_{e_1} \leq c_{e_9} \text{ and } c_{e_1} \leq c_{e_{14}}.$$

Note that the ordering of the variables has an impact on the generated constraints. Indeed, for the same graph, we give on the rightside of Figure 6.4 a different labelling, where $(v_1, v_4, v_7)(v_2, v_5, v_8)(v_3, v_6, v_9)$ translates into

$$(c_{e_1}, c_{e_2}, c_{e_3})(c_{e_4}, c_{e_5}, c_{e_6})(c_{e_7}, c_{e_8}, c_{e_9})(c_{e_{10}}, c_{e_{11}}, c_{e_{12}})(c_{e_{13}}, c_{e_{14}}, c_{e_{15}}).$$

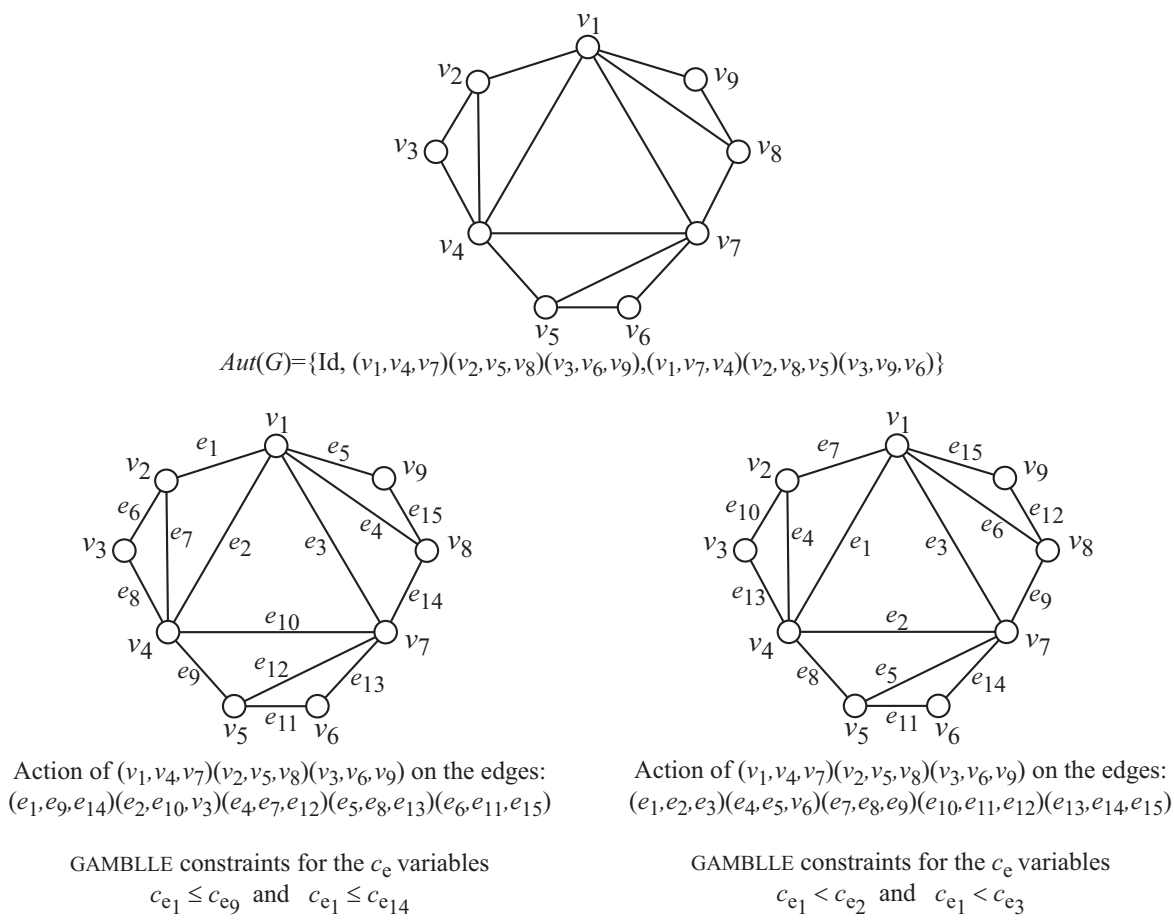
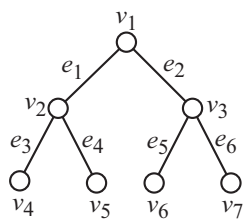


Figure 6.4 Illustration of the generation of GAMBLLE constraints.

The cycle with smallest index is then $C = (c_{e_1}, c_{e_2}, c_{e_3})$ and we therefore obtain the following strenghtened constraints:

$$c_{e_1} < c_{e_2} \text{ and } c_{e_1} < c_{e_3}.$$

As a second example, consider the complete binary tree of Figure 6.5. NAUTY produces permutations $(v_2, v_3)(v_4, v_6)(v_5, v_7), (v_4, v_5)$ and (v_6, v_7) of the vertex set which correspond to permutations $(e_1, e_2)(e_3, e_5)(e_4, e_6), (e_3, e_4)$ and (e_4, e_6) of the edges when they are ordered as shown on the leftside of Figure 6.5, according to the lexicographical ordering of their pair of endvertices. The GAMBLLE constraints associated with the c_e variables are $c_{e_1} < c_{e_2}$, $c_{e_3} < c_{e_4}$ and $c_{e_5} < c_{e_6}$, respectively. The reverse ordering of the edges gives different GAMBLLE constraints since one of them is not a strict inequality. Indeed, the three permutations of the vertex set translate into permutations $(e_1, e_3)(e_2, e_4)(e_5, e_6), (e_3, e_4)$ and (e_1, e_2) of the edge set, and the associated GAMBLLE constraints are $c_{e_1} \le c_{e_3}$, $c_{e_3} < c_{e_4}$ and $c_{e_1} < c_{e_2}$, respectively.



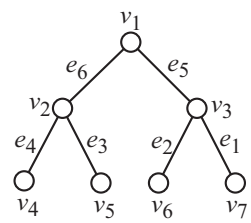
$(v_2,v_3)(v_4,v_6)(v_5,v_7)$, (v_4,v_5), (v_6,v_7)

translate into

$(e_1,e_2)(e_3,e_5)(e_4,e_6)$, (e_3,e_4), (e_5,e_6)

GAMBLLE constraints for the c_e variables

$c_{e_1} < c_{e_2}$, $c_{e_3} < c_{e_4}$ and $c_{e_5} < c_{e_6}$



$(v_2,v_3)(v_4,v_6)(v_5,v_7)$, (v_4,v_5), (v_6,v_7)

translate into

$(e_1,e_3)(e_2,e_4)(e_5,e_6)$, (e_3,e_4), (e_1,e_2)

GAMBLLE constraints for the c_e variables

$c_{e_1} \leq c_{e_3}$, $c_{e_3} < c_{e_4}$ and $c_{e_5} < c_{e_6}$

Figure 6.5 GAMBLLE constraints for two different edge orderings of a binary tree.

In the experiments reported in the next section, we consider all permutations obtained using NAUTY and CLUSTERS, and the edges are ordered according to the lexicographical ordering of their pair of endvertices. For example, consider again the P_4 of Figure 6.1(a). CLUSTERS does not generate any permutation since the graph does not contain any stable set of twins or clique of twins. NAUTY generates permutation $(v_1, v_4)(v_2, v_3)$ that translates into permutation $(\underline{c}_{v_1}, \underline{c}_{v_4})(\underline{c}_{v_2}, \underline{c}_{v_3})$ of the $\underline{c}_v$ variables. The considered cycle C is therefore $(\underline{c}_{v_1}, \underline{c}_{v_4})$ which gives the GAMBLLE constraint $\underline{c}_{v_1} \leq \underline{c}_{v_4}$ that forbids solution s_4. Similarly, with the $\overline{c}_v$ variables, we get the GAMBLLE constraint $\overline{c}_{v_1} \leq \overline{c}_{v_4}$ which forbids solution s_4. For the d_v variables, we get the GAMBLLE constraint $d_{v_1} \leq d_{v_4}$ that does not break any symmetry since $d_{v_1} = d_{v_4} = 0$ in both s_3 and s_4. The permutation of the c_e variables associated with $(v_1, v_4)(v_2, v_3)$ is (c_{e_1}, c_{e_3}), and the associated GAMBLLE constraint $c_{e_1} \leq c_{e_3}$ again forbids s_4.

Consider now the clique K_3 of Figure 6.1(b). Both CLUSTERS and NAUTY produce permutations (v_1, v_2) and (v_2, v_3). The corresponding GAMBLLE constraints for the $\underline{c}_v$ variables are $\underline{c}_{v_1} \leq \underline{c}_{v_2} \leq \underline{c}_{v_3}$, which remove all optimal solutions except s_1 and s_2. The GAMBLLE constraints for the $\overline{c}_v$ variables are $\overline{c}_{v_1} \leq \overline{c}_{v_2} \leq \overline{c}_{v_3}$ which are only satisfied by solutions s_1 and s_4. With the d_v variables, we get $d_{v_1} \leq d_{v_2} \leq d_{v_3}$ satisfied by s_4 and s_5. The only variables that leave exactly one solution among the six equivalent ones are the c_e variables. Indeed, their associated GAMBLLE constraints are $c_{e_1} < c_{e_2} < c_{e_3}$ which forbid all optimal solutions except s_1. Note that GAMBLLE constraints associated with different sets of permutable variables cannot be combined, since all solutions to the minimum deficiency problem are then possibly forbidden. For example, for the clique K_3, the union of the GAMBLLE constraints $d_{v_1} \leq d_{v_2} \leq d_{v_3}$ and $c_{e_1} < c_{e_2} < c_{e_3}$ forbids all solutions: $d_{v_2} \leq d_{v_3}$ and $c_{e_3} > \max\{c_{e_1}, c_{e_2}\}$ is equivalent to $c_{e_3} - c_{e_1} \leq c_{e_3} - c_{e_2}$ which implies $c_{e_2} \leq c_{e_1}$.

6.6 Computational experiments

6.6.1 Experimental setup

To generate GAMBLLE constraints, two parameters come into play. The first one is the method to obtain a set of permutations, either through the NAUTY library (`N`) or the CLUSTERS method (`C`). The second is the family of permutable variables used, which can be the color c_e of the edges (`col`), the minimum color $\underline{c}_v$ at the vertices (`min`), the maximum color $\overline{c}_v$ at the vertices (`max`), or the deficiency d_v at the vertices (`def`). The set of extra constraints and the resulting algorithm when using them are both denoted by putting the letters of the options together in the format `G<method><family>`. Also, `none` denotes the original model solved without any GAMBLLE constraints.

The following two datasets are considered in our experiment: **D1**, the complete set of connected simple graphs of size 4 to 9, and **D2**, a series of random connected simple graphs, 8 for each pair (n, p) with $4 \leq n \leq 100$ vertices and density in $(p - 0.05, p + 0.05]$ with $p \in \{0.1, 0.2, \ldots, 0.9\}$.

The tests were run on a Lenovo Thinkpad X300 laptop, with Intel Core 2 Duo CPU at 1.2 GHz and 4Gb of RAM. Given a graph and a family of permutable variables, the pre-processing step generates the GAMBLLE constraints, using either NAUTY (v2.4r2) or CLUSTERS. Using IBM/ILOG's optimization suite, the basic model is expressed in *OPL* (Optimization Programming Language). It is then instanciated with the graph and augmented with the additional constraints. This object is solved with either *cp optimizer* (v12.2) to competitively find the deficiency, or *CPLEX* (v12.2) to find the list of optimal solutions. For the latter the generation of graph images in post-processing relies on *Graphviz*. This whole process and the batch processing are both orchestrated by programs written in *Ruby*.

6.6.2 Automorphism group classes

Given a graph G, it may happen that $CAut(G)$ is strictly contained in $Aut(G)$. Also, we possibly have $CAut(G) = \{Id\}$, which means that G does not contain any stable set of twins or clique of twins. We distinguish the following four cases, denoted A1, A2, A3 and A4, defined in Table 6.1.

The graph $K_{2,3}$ of Figure 6.3 is in A2 since both CLUSTERS and NAUTY produce the 3 generators for the 12 permutations in $Aut(K_{2,3})$. The graph in Figure 6.4 belong to A3 since CLUSTERS does not produce any permutation while NAUTY does. The graph in Figure 6.2 is in A4 since NAUTY produces permutation $(v_5, v_7)(v_6, v_8)$ that does not belong to $CAut(G)$

Table 6.1 Automorphism group classes.

	$1 = \lvert CAut(G)\rvert$	$1 < \lvert CAut(G)\rvert$
$CAut(G) = Aut(G)$	A1 `GC<fam>` $\equiv$ `GN<fam>` $\equiv$ `none`	A2 `GC<fam>` $\equiv$ `GN<fam>`
$CAut(G) \subsetneq Aut(G)$	A3 `GC<fam>` $\equiv$ `none`	A4

and $|CAut(G)| = 24$. An example of graph in A1, is shown in Figure 6.6.

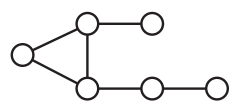


Figure 6.6 The smallest graph G with $CAut(G) = Aut(G) = \{Id\}$.

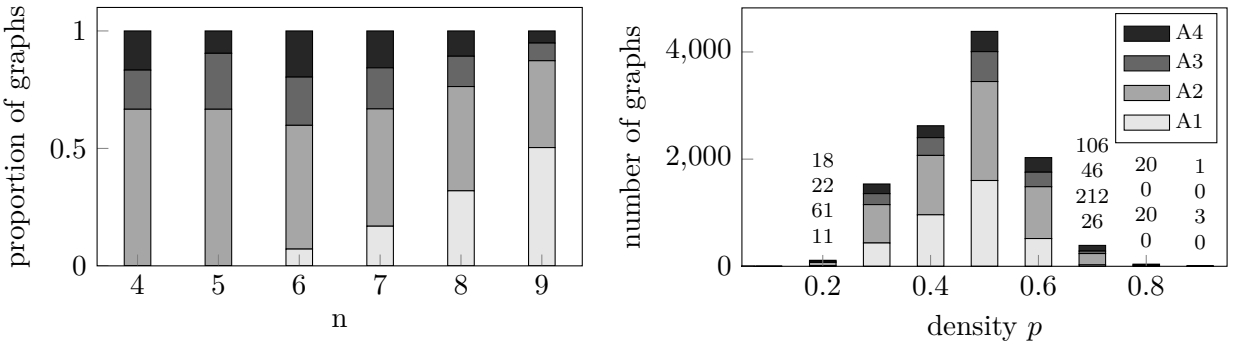
When G is in class A1 or A3, `GC<family>` is equivalent to `none`, and when G is in A1 or A2, `GC<family>` and `GN<family>` produce the same results.

As shown in Figure 6.7, the proportion of graphs in **D1** that belong to A1 increases monotonically with the the number of vertices. This becomes even more clear in Figure 6.8 for the random graphs in **D2**. When $|Aut(G)| > 1$, class A2 seems to dominate, which suggests that most automorphisms are due to stable sets of twins and cliques of twins.

In **D1**, when fixing the number n of vertices and varying the density d, the middle range of density tends to have a bigger proportion of graphs with no automorphism, contrary to the extremes that mostly have some. Figure 6.7 shows the case of $n = 8$. This observation seems to stay true for larger graphs (see Figure 6.8 with $n = 20$).

Let $g_N(G)$ (respectiveley $g_C(G)$) denote the number of generators produced by NAUTY (respectively CLUSTERS) when applied to G. Figures 6.7 and 6.8 clearly show that most graphs belong to A1 and A2, in which case we have $g_N(G) = g_C(G)$. If G belongs to A3 or A4, $g_C(G)$ tends to be usually only slightly smaller than $g_N(G)$, as shown in Table 6.2, where we indicate the number of graphs G with $n = 8$ vertices for every pair $(g_N(G), g_C(G))$. Indeed among the 11117 graphs, only 228 of them (i.e., 2%) have $g_N(G) - g_C(G) > 1$. By summing the numbers on the diagonal, we obtain that 8565 graphs out of 11117 (i.e. 77%) belong to A1 or A2. These observations justify the use of CLUSTERS since $CAut(G) = Aut(G)$ in a majority of cases.

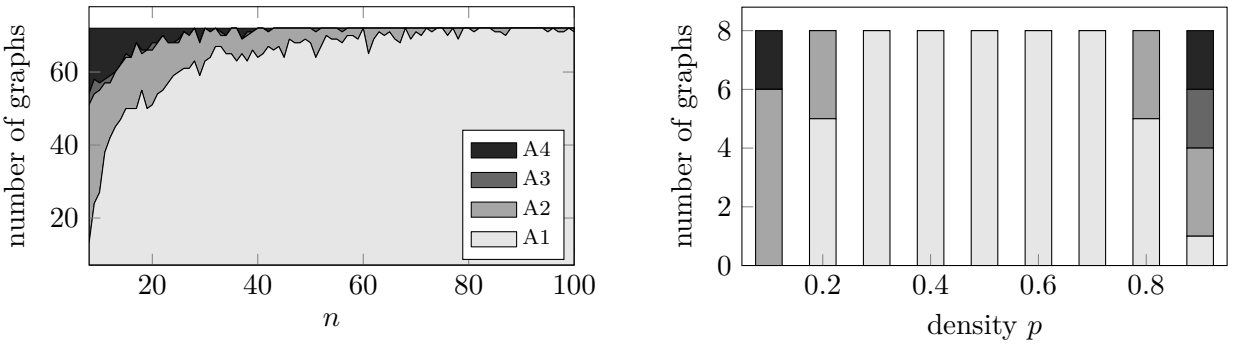
Using again data set **D1**, we analyze in Table 6.3 the relationship between the minimum deficiency $d(G)$ and the automorphism group class $a(G) \in \{$A1,A2,A3,A4$\}$ to which G belongs,



(a) Distribution for every $n \in \{4, 5, \ldots, 9\}$.

(b) Distribution for $n = 8$ and $p \in \{0.1, 0.2, \ldots, 0.9\}$.

Figure 6.7 Distribution of the automorphism group classes for dataset **D1**.



(a) Distribution for every $n \in \{4, 5, \ldots, 100\}$.

(b) Distribution for $n = 20$ and $p \in \{0.1, 0.2, \ldots, 0.9\}$.

Figure 6.8 Distribution of the automorphism group classes for dataset **D2**.

as well as the relationship between $d(G)$ and the number $g_N(G)$ of generators produced by NAUTY. We indicate the percentage of graphs with $n = 8$ vertices for every pair $(d(G), g_N(G))$ and every pair $(d(G), a(G))$. Notice first that most graphs (97.7%) have minimum deficiency 0. Among the 257 graphs (2.31%) with $d(G) > 0$, only 6 of them (0.05 % of 11117, which is also 2.33% of 257) have no automorphism (i.e., belong to A1). Also, it appears that the average value of $g_N(G)$ tends to increase when the minimum deficiency increases : it is equal to 1.11 for graphs with $d(G) = 0$, to 2.39 for graphs with $d(G) = 1$, and to 4.5 for graphs with $d(G) = 2$.

Table 6.2 $g_N(G)$ versus $g_C(G)$ for $n = 8$

$g_C(G)$ \ $g_N(G)$	0	1	2	3	4	5	6	7
7								1
6							8	1
5						39	4	
4					177	22	1	1
3				555	71	1		
2			1408	263	14			
1		2825	690	48	2			
0	3552	1273	144	16	1			

Table 6.3 $d(G)$ versus $g_N(G)$ and the automorphism group classes.

$d(G)$	$g_N(G)$								automorphism class			
	0	1	2	3	4	5	6	7	1	2	3	4
0	31.9	36.34	19.46	7.33	2.1	0.44	0.1	0.03	31.9	43.28	12.48	10.04
1	0.05	0.52	0.71	0.6	0.27	0.1	0.02		0.05	1.08	0.42	0.72
2					0.02	0.02				0.03		0.01

6.6.3 Comparison of algorithms

Let $L_{\texttt{none}}(G)$ be the set of optimal solutions to the minimum deficiency problem when no extra constraints are added to the constrained programming model of Section 6.2. Such a set can be represented by a graph $H_{\texttt{none}}(G)$ where each vertex is a solution in $L_{\texttt{none}}(G)$, and two solutions are adjacent if one can be transformed into the other by swapping the two colors along a path or cycle of alternating colors. These are the two most fundamental neighborhoods used in heuristics for solving the minimum deficiency problem (Bouchard et al., 2009). They leave unaltered the deficiency in their interior vertices, so there can only be a potential effect on the deficiency at the two ends of a path. Consider for example the graph in Figure 6.9. The coloring on the leftside is optimal since the deficiency is zero. A swapping of colors 1 and 2 on the cycle containing vertices v_1, v_2, v_5, v_6 produces a new optimal solution. These two solutions solutions are equivalent, one being obtained from the other by permuting vertices v_2 and v_5 which belong to a stable set of twins. The two solutions are therefore linked by an edge in $H_{\texttt{none}}(G)$. A swapping of colors 1 and 2 on a path is shown on the rightside of Figure 6.9. In this case, we obtain a deficiency at vertex v_8. Hence, this third coloring is not optimal and therefore not represented in $H_{\texttt{none}}(G)$.

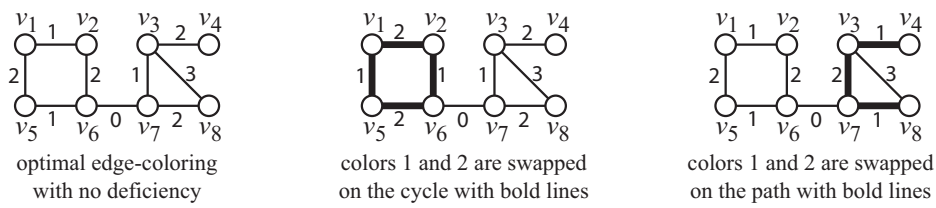


Figure 6.9 Illustration of swappings on cycles and paths.

Let `alg` be any of the proposed algorithms. Since `alg` adds constraints to the original constrained programming model, it ideally forbids some solutions in $L_{\texttt{none}}(G)$ to produce a subset $L_{\texttt{alg}}(G)$ of optimal solutions. Therefore, the graph $H_{\texttt{alg}}(G)$ representing the links between the optimal solutions when the extra constraints produced by `alg` are taken into account is always an induced subgraph of the original graph $H_{\texttt{none}}(G)$.

In the following representations of $H_{\texttt{alg}}(G)$, a solid edge corresponds to a swapping on a path and a dashed line to a swapping on a cycle, and its width is proportional to the number of vertices in the path or cycle.

One way to compare the performance of the algorithms is to observe their impact of their respective extra constraints on the solution space, and in particular on the the sets $L_{\texttt{alg}}(G)$ of optimal solutions. In the best case, a set of symmetry breaking constraints can at most divide the size of the original optimal solution space by $|Aut(G)|$. Hence,

$$\frac{|L_{\texttt{none}}(G)|}{|Aut(G)|} \leq |L_{\texttt{alg}}(G)| \leq |L_{\texttt{none}}(G)|.$$

These bounds are not tight. Indeed, consider for example the P_4 in Figure 6.1(a). It has two automorphisms, the identity permutation and $(v_1\ v_4)(v_2\ v_3)$, and four optimal solutions. As already mentioned, the last solution is equivalent to the third one and is removed by our algorithms. Hence, every proposed algorithm `alg` breaks all symmetries while we have

$$\frac{|L_{\texttt{none}}(P_4)|}{|Aut(P_4)|} = \frac{4}{2} < 3 = |L_{\texttt{alg}}(P_4)| < 4 = |L_{\texttt{none}}(P_4)|.$$

This means that when the lower bound is reached, we have the guarantee that all symmetries due to graph automorphisms have been broken, which is our objective. But it is also possible to attain that goal and still not reach the lower bound.

We first consider the four graphs of Figure 6.10 with five vertices. The first one is the $K_{2,3}$ of Figure 6.3. The second one is a clique on 5 vertices, and hence a clique of twins. The third

one is obtained from G_2 by removing one edge, and contains a clique of twins and a stable set of twins. The fourth one is obtained from G_2 by deleting two disjoint edges, and contains two stable sets of twins. The results of our algorithms are shown in Table 6.4. On the left part of the table, we indicate for every graph G_i the class to which its belong (i.e., A1, A2, A3 or A4), the sizes of the cluster-automorphism $CAut(G_i)$ and of the automorphism group $Aut(G_i)$, and the minimum deficiency $d(G_i)$. On the rightside of the Table, we indicate the size of the sets $L_{\texttt{alg}}(G_i)$. The optimal solution spaces $H_{\texttt{alg}}(G_i)$ are shown in Figures 6.11, 6.12, 6.13 and 6.14.

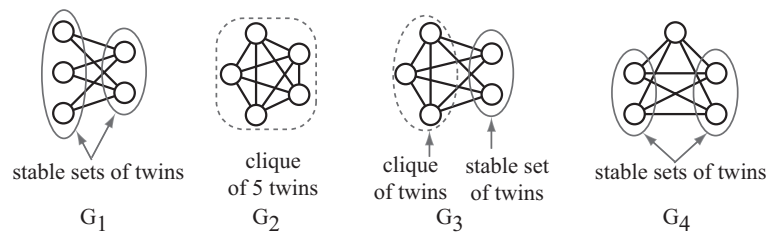


Figure 6.10 Four graphs with 5 vertices.

For the first graph $G_1 = K_{2,3}$, the 12 original optimal solutions are all permutations of the same one, and three of the families of permutable variables are able to break all symmetries, as reaching the bound $\frac{|L_{\texttt{none}}(G_1)|}{|Aut(G_1)|} = \frac{12}{12} = 1$ gives us that guarantee. Only `G{C,N}def` is not able to break all symmetries. Graphs G_2, G_3 and G_4, help to illustrate how the removal of an edge can change the landscape of the solution space dramatically. Note that the bound $\frac{|L_{\texttt{none}}(G_3)|}{|Aut(G_3)|} = \frac{96}{12} = 8$ is reached for G_3 with `G{C,N}col`. The fourth graph G_4 illustrates a case where $1 < |CAut(G)| < |Aut(G)|$. Predictably, the NAUTY method gives better results than CLUSTERS.

The `col` family of permutable variables is clearly the most efficient one for breaking symmetries. This is probably because it is the only one constraining the variables representing the color on an edge (the key decision variables of our problem), as well as the only one that has sometimes strengthened constraints. On the opposite end, using the `def` family of

Table 6.4 Size of the optimal solution spaces for the proposed algorithms.

graph	class	$\lvert CAut(G_i)\rvert$	$\lvert Aut(G_i)\rvert$	$d(G_i)$	none	G_col		G_min		G_max		G_def	
						C	N	C	N	C	N	C	N
G_1	A2	12	12	0	12	1	1	1	1	1	1	12	12
G_2	A2	120	120	2	720	18	18	24	24	24	24	96	96
G_3	A2	12	12	1	96	8	8	22	22	22	22	32	32
G_4	A4	4	8	1	48	16	8	12	10	16	8	32	32

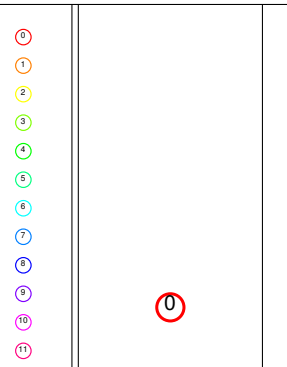
none	G{C,N}col	G{C,N}min	G{C,N}max	G{C,N}def
				
12	1	1	1	12

Figure 6.11 Optimal solutions spaces $H_{\texttt{alg}}(G_1)$ for G_1.

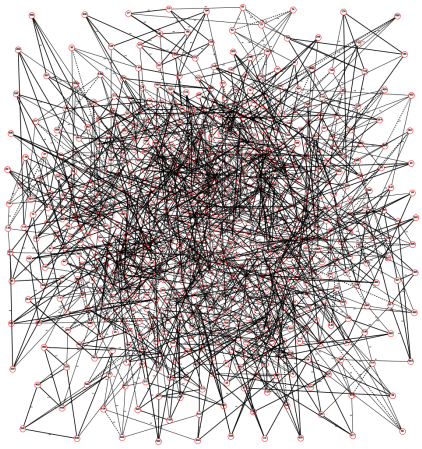
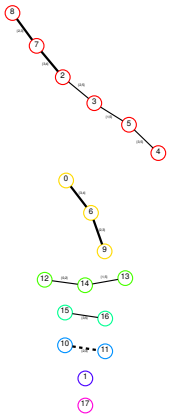
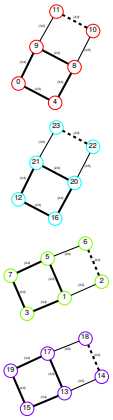
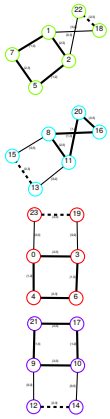
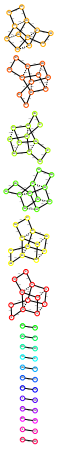
none	G{C,N}col	G{C,N}min	G{C,N}max	G{C,N}def
				
720	18	24	24	96

Figure 6.12 Optimal solution spaces $H_{\texttt{alg}}(G_2)$ for G_2.

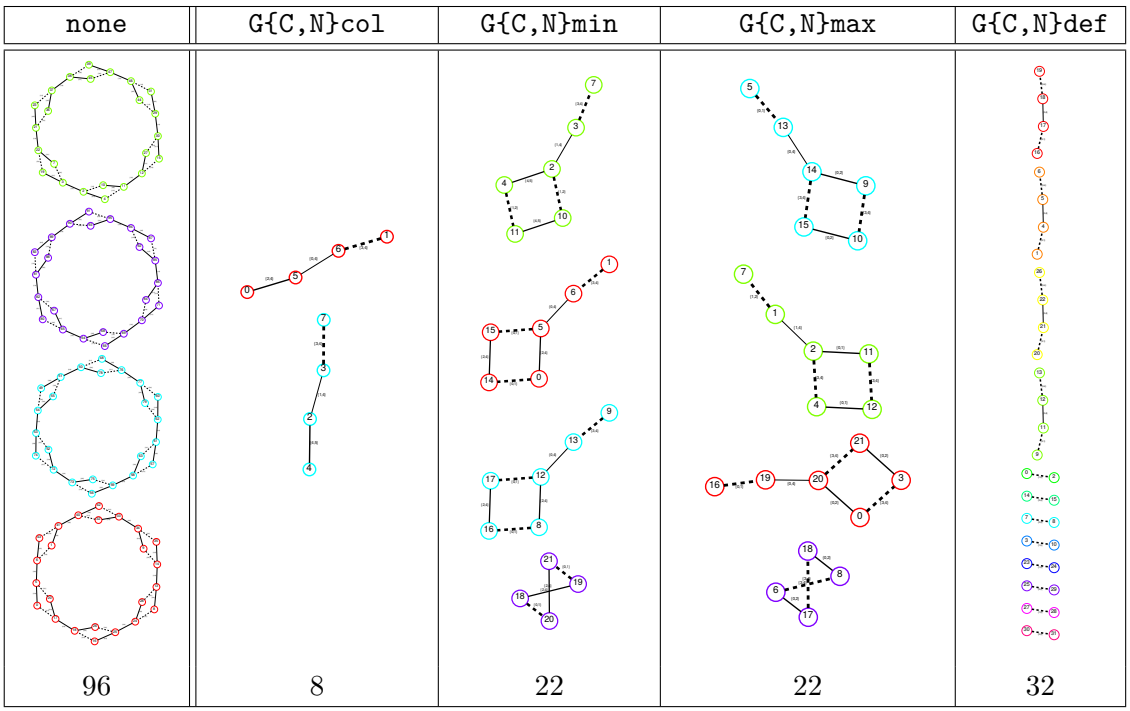


Figure 6.13 Optimal solution spaces $H_{\texttt{alg}}(G_3)$ for G_3.

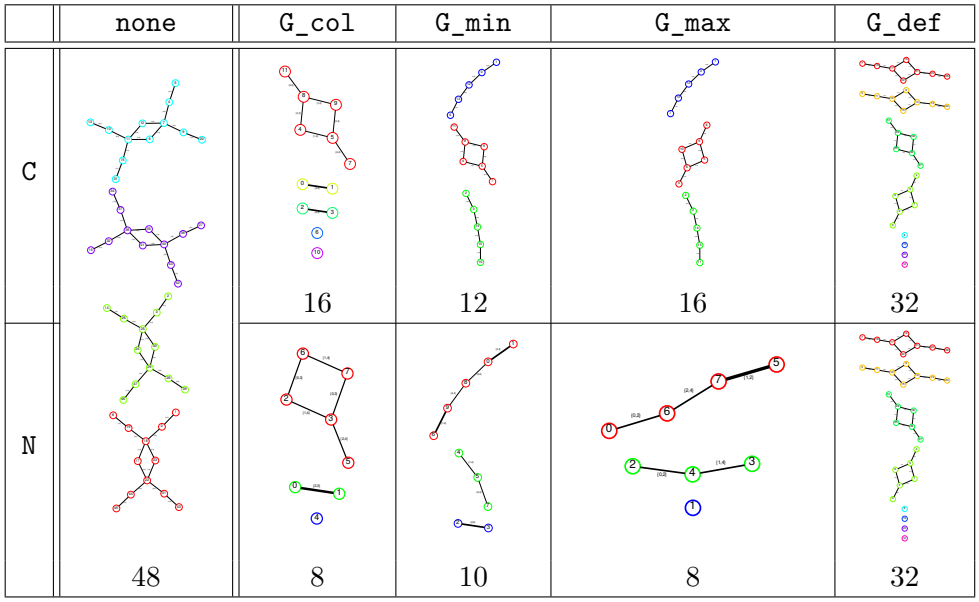


Figure 6.14 Optimal solution spaces $H_{\texttt{alg}}(G_4)$ for G_4.

permutable variables has a much smaller impact on the reduction of the optimal solution space. The other two families `min` and `max` seem to have comparable performances, between the aforementioned two extremes.

On a final note, we have to mention that this approach of explicit observation of the optimal solution spaces is unfortunately very limited. Only extremely simple graphs can be considered, usually solved efficiently in less than 0.02 seconds, but whose enumeration and post-processing can still take up to a couple of hours. Also, the size of an optimal solution space correlates to the time it takes CPLEX to enumerate all its solution using the MIP model.

Six larger graphs are considered for comparing the computing times needed to solve the constrained programming model with or without the extra constraints. The six graphs $G_5, \ldots, G_{10}$ are shown in Figure 6.15. G_5 and G_6 are cliques with 6 and 7 vertices, respectively. G_7 is obtained from G_6 by removing one edge. G_8 (respectively G_9) is obtained from G_6 by removing two incident (respectively disjoint) edges, while G_{10} is obtained from G_6 by removing 3 disjoint edges. These graphs contain cliques of twins shown using boxes with dashed lines, and stable sets of twins shown using boxes with plain lines.

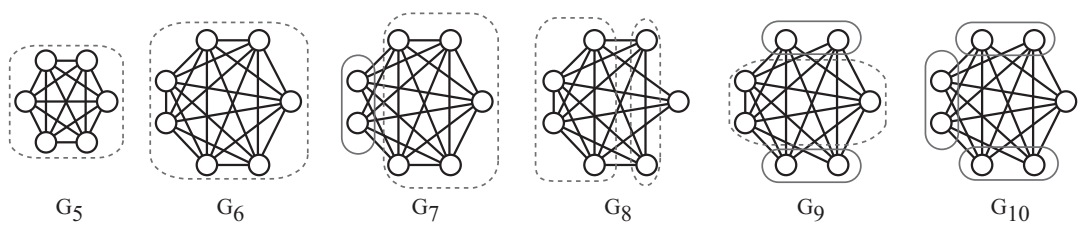


Figure 6.15 Six graphs with 6 and 7 vertices.

Table 6.5 Computing times (in seconds) for six graphs.

graph	class	$\lvert CAut(G_i)\rvert$	$\lvert Aut(G_i)\rvert$	$d(G_i)$	`none`	`G_col`		`G_min`		`G_max`		`G_def`	
						`C`	`N`	`C`	`N`	`C`	`N`	`C`	`N`
G_5	A2	720	720	0	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
G_6	A2	5040	5040	3	>7200	11.53	11.53	415.03	415.03	791.37	791.37	1631.00	1631.00
G_7	A2	240	240	2	298.32	2.87	2.87	91.21	91.21	93.13	93.13	177.56	177.56
G_8	A2	48	48	1	7.27	0.26	0.26	3.63	3.63	3.68	3.68	10.06	10.06
G_9	A4	24	48	1	11.58	0.58	0.37	8.82	7.63	8.90	6.48	12.20	12.94
G_{10}	A4	8	48	2	34.69	7.96	2.60	21.15	5.29	18.74	7.73	31.38	30.58

All our previous general observations on the relative performance of the algorithms for the size of the optimal solution space remain true in the case of computing times, except that the methods based on the `min` family of permutable variables are slightly superior to those with the `max` family. Again, `col` gives the best results, with computing times reduced by at least one order of magnitude. Also, NAUTY consistently outperforms CLUSTERS, but

within the same order of magnitude. Thus, `GNcol` clearly emerges as the best algorithm, combining successfully two ideas, the choice of the family of permutable variables and the method for finding a set of generators for the automorphism group. The graph with larger automorphism group is the clique G_6 on seven vertices with $|Aut(G_6)| = 5040$. While the original constrained programming model does not find a proven optimal solution in 2 hours of computation, `GNcol` solves the problem in 11 seconds.

The impact of the various methods on the computing time is also shown in Table 6.6 where we analyze the total time needed to solve the minimum deficiency problem for all graphs with $n = 4, 5, 6, 7, 8$ vertices. We do not report the results for $n = 9$ since among the 261084 graphs with 9 vertices, the CP solver has not produced proven optimal solutions within 2 hours of computation for about 100 of them.

Table 6.6 Total computing times (in seconds) to solve all the graphs with $4, 5, 6, 7$ and 8 vertices

n	none	G_col		G_min		G_max		G_def	
		C	N	C	N	C	N	C	N
4	<0.01	<0.01	0.01	0.02	0.01	0.03	0.03	0.03	0.02
5	0.12	0.06	0.06	0.07	0.08	0.1	0.1	0.12	0.12
6	0.3	0.34	0.35	0.42	0.5	0.48	0.47	0.43	0.43
7	4403.9	22.1	17.2	327.8	314.42	541.0	530.8	1092.3	1092.7
8	4702.9	249.4	203.4	1406.6	1290.8	1529.9	1415.3	4526.6	4546.8

The orientation of the *less than or equal* ($\leq$) inequality constraints used in section 6.5 to generate GAMBLLE constraints has an impact on the performance of the algorithms, as it interacts with both the constraint that forces the usage of color 0, and the rule to choose the variable with the smallest index. Also, as already mentioned, the smallest index rule makes the GAMBLLE constraints very sensitive to the labelling of the edges for the algorithms based on the `col` family of permutable variables. This is shown in Table 6.7 where we compare three different settings for graph G_9. The first case (called $G_9(\leq)$) is the original one, where the vertices are labelled as shown at the leftside of Figure 6.16, the edges are labelled according to the lexicographical ordering of their pair of endvertices, and a cyclic permutation C with smallest indexed variable u_i leads to inequalities $u_i \leq u_j$ or $u_i < u_j$ for all u_j in C with $j \neq i$. The second test (called $G_9(\geq)$) uses inequalities $u_i \geq u_j$ or $u_i > u_j$ instead of the $\leq$ or $<$ orignal ones. The third test (called G_9(REV)) uses the $\leq$ or $<$ inequalities, but considers the reverse labelling of the vertices with the corresponding labelling of the edges (according to the lexicographical ordering of their pair of endvertices) as shown at the rightside of Figure 6.16. While the models remain valid with these changes, we observe that the performance may significantly drop when we do not use the original setting. For example, the problem is solved in 7 seconds with the original settings, while the use of the $\geq$ or $>$

inequalities increases the computing time to 100 seconds.

Table 6.7 Computing times for three variants of the proposed algorithms applied to G_9.

graph	none	G_col		G_min		G_max		G_def	
		C	N	C	N	C	N	C	N
G_9 ($\leq$)	11.58	0.58	0.37	8.82	7.63	8.90	6.48	12.20	12.94
G_9 ($\geq$)	11.58	6.48	4.04	116.18	100.15	94.93	51.41	14.13	12.77
G_9 REV	17.20	2.75	1.24	12.33	10.96	11.45	10.28	39.52	40.28

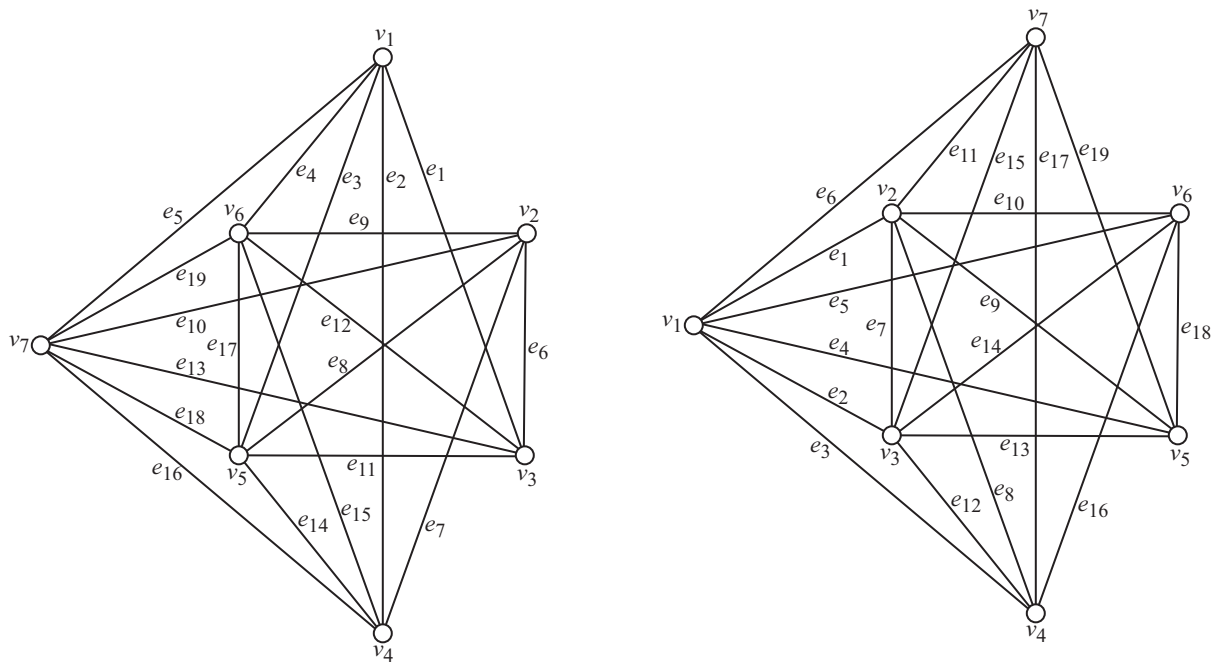


Figure 6.16 Two labellings of the vertices and of the edges of G_9.

6.7 Conclusion

The generation of GAMBLLE constraints is a general technique to help solving CP models of graph optimization problems. In the present paper, its potency is demonstrated through an application to the minimum deficiency problem.

A straightforward CP model to find the exact deficiency is hindered by an overwhelming number of equivalent optimal colorings, in part due to the automorphisms of the considered graph. When included in the model, the GAMBLLE constraints help to cut down the solution space by forbidding some equivalent optimal solutions, and thus improve the time performance of the solver. The total number of extra constraints generated remains very

small, since it only depends on the number of generators found for the automorphism group, which is in the order of n in the worst case. In our tests, the best proposed algorithm `GNcol` improves by at least one order of magnitude the basic model. It is particularly efficient for graphs that have a lot of symmetries.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Nous discutons ici de notre contribution scientifique globale, d'un point de vue théorique et appliqué.

7.1 Synthèse des travaux

Au niveau théorique, bien que l'existence d'une réduction du k-LCCP au k-CCCP ne soit pas surprenante (comme les deux problèmes sont $\mathcal{NP}$-complets), la réduction que nous proposons n'augmente pas le nombre k de couleurs, contrairement aux transformations préexistantes. Aussi, les outils créés pour réaliser cette réduction offrent des propriétés nouvelles et fort intéressantes.

Pour le nombre maximal de couleurs nécessaires pour obtenir une coloration avec déficience optimale $S(G)$ d'un graphe G, nous avons proposé une borne supérieure dépendant du nombre de sommets n et de la déficience $d(G)$ du graphe

$$S(G) \leq 2n - 4 + d(G).$$

Celle-ci remplace avantageusement la borne triviale utilisée précédemment qui était m. Elle se base sur et étend en même temps une borne pré-existante ne s'appliquant qu'au cas sans déficience.

Au niveau des applications, en combinant notre borne à la conjecture que $d(G) \leq n$, nous obtenons que $3n - 4$ couleurs devraient être suffisantes pour garder toutes les solutions optimales dans une modélisation qui requiert de spécifier un nombre fixe de couleurs. Si un graphe contredisant la conjecture était amené à être introduit dans l'un de nos modèles, par construction, il serait automatiquement détecté. Il fournirait alors à la fois un contre-exemple précieux à la conjecture, et un nouveau nombre de couleurs à utiliser, qui permettrait de relancer la résolution à nouveau avec un nombre de couleurs garanti suffisant cette fois-ci.

Cette borne pragmatique nous a permis de proposer des modélisations en Programmation en Nombre Entiers et en Programmation par Contraintes avec un nombre raisonnable de variables et de contraintes.

Pour la Programmation par Contraintes, nous avons poussé la recherche plus loin et travaillé à casser les symétries inhérentes au problème et à sa modélisation. Nous avons développé les contraintes de type GAMBLLE, qui brisent en général une majeure partie de la symétrie

engendrée par les automorphismes de graphe, de manière heuristique et particulièrement économe en nombre de contraintes rajoutées, en général plus petit que n.

Afin de mesurer la performances de ces nouvelles contraintes, nous avons fait un nombre extensif de mesures, et notamment développé un outil autant pratique que visuellement plaisant pour illustrer leur impact sur la réduction effective de l'espace de solutions optimales, comme espéré par le cassage de symétrie.

Au final, à l'aide de tous le travail accompli, nous avons réussi à déterminer de manière exacte la déficience pour la quasi-entièreté des 273189 graphes simples connexes de taille entre 4 et 9 inclus, à l'exception de moins d'une centaine. Ceci nous a permis d'explorer la relation entre la déficience et d'autres invariants fondamentaux comme n, m, la densité, et la taille du groupe d'automorphisme $Aut(G)$.

7.2 Limitations des solutions proposées

Par la construction même de nos outils, notre réduction du k-LCCP au k-CCCP est limitée aux $k \geq 12$. Les k inférieurs à 12 doivent eux être ramenés à un problème cyclique à 12 couleurs. Pour l'instant, il ne nous apparaît pas de moyen évident pour adapter et étendre notre réduction à tous les k.

Quant à nos modélisations, celles en Programmation en Nombres Entiers s'avèrent décevantes en termes de performance, principalement à cause de la myriade de solutions optimales équivalentes symétriques, engendrées en partie par la structure du problème et les automorphismes des graphes. Le modèle basé sur la Programmation par Contraintes, bien que clairement plus efficace, est lui aussi affecté par ces nombreuses solutions optimales.

Même lorsque nous introduisons les contraintes GAMBLLE, conçues spécifiquement pour attaquer cette difficulté, nous obtenons un gain très significatif de performance, mais pas encore satisfaisant. En effet, au-delà de 10 sommets, les graphes particulièrement denses continuent à poser de plus en plus de difficultés, et la proportion de graphes aléatoires avec des automorphismes diminue progressivement, rendant par là le cassage de symétrie du graphe moins pertinent, sauf pour certains cas particuliers. Heureusement, en pratique, soit les graphes rencontrés sont construits de manière théorique, soit ils modélisent des situations issues du monde réel. Dans les deux cas, cela donne souvent lieu à une certaine structure dans le graphe, qui va générer de la symétrie que nos contraintes peuvent travailler à briser.

7.3 Améliorations futures

Nos outils de réduction permettent de transformer un problème linéaire avec des couleurs interdites ou imposées en un problème cyclique sans contraintes supplémentaires. Ceci permet d'ouvrir tout un champ de variations du problème original linéaire, avec des contraintes possiblement très spécifiques ou ciblées, à une modélisation toute simple cyclique.

Aussi, le principe général de nos contraintes GAMBLLE peut dans certains cas facilement être adapté aux modélisations d'autres invariants de graphes, dont le calcul est également sévèrement affecté par les automorphismes de graphe. Il serait certainement intéressant de formellement spécifier les propriétés que doit avoir un invariant pour qu'il soit possible d'y intégrer nos contraintes, et d'observer si les fantastiques performances dans le cas particulier du calcul de la déficience se maintiennent en général.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Dans le cadre de cette thèse, nous nous sommes intéressé à un problème de confection d'horaire, dont l'origine est ancrée dans des problèmes rencontrés dans le monde industriel. En effet, il est souvent souhaitable en pratique de compacter l'horaire d'utilisations des différents éléments d'un système. Il peut par exemple s'agir de machines qu'il est coûteux de faire tourner à vide ou maintenir en fonction entre deux utilisations, ou bien de travailleurs qui se retrouvent à devoir attendre entre des tâches à accomplir, sans pouvoir être productifs.

Notre premier intérêt à été de mettre en relation deux types d'horaire, l'un limité dans le temps et l'autre amené à se répéter périodiquement, et de montrer leur similitude en terme de complexité.

Une fois cette étude théorique menée à bien, nous nous sommes concentrés sur l'aspect plus pratique de développer un algorithme pour résoudre de manière efficace le problème de la déficience. Celui-ci consiste dans son essence à minimiser le nombre total de trous dans l'horaire du problème modélisé. Notre recherche nous a amené tout d'abord à explorer plusieurs pistes, car aucune ne semblait être prometteuse, pour ensuite peaufiner celle de la Programmation par Contraintes. Pour ce faire, nous nous sommes intéressés à l'impact des symétries d'un graphe sur la résolution d'un modèle qui y est associé. Nous avons ainsi développé un nouveau type de contraintes heuristiques appelées GAMBLLE, extrêmement intuitives, d'une rare simplicité et rapidité à générer, économes en nombre, et finalement d'une impressionnante performance pour notre problème.

Au terme de ce travail, nous constatons avec fierté et un soupir de soulagement tout le travail accompli, souvent ardu et démoralisant, qui nous a amené à réduire de plusieurs heures à moins d'une seconde le temps de calcul de la déficience des graphes de petite taille. Entre le modèle évident qui vient tout d'abord à l'esprit et la version finale débordant d'inventivité et d'astuces, il y a eu plusieurs étapes clés. Chacune a amené son lot de doutes, douleurs, déceptions, découragements. Il s'en est plusieurs fois tenu à un cheveu que chacune soit la dernière. Et pourtant, à force de serrer les dents et de ne pas lâcher le morceau, et étant également entouré d'une fabuleuse équipe, à chaque fois les difficultés ont finalement été surmontées, un nouvel ordre de grandeur de temps a été gagné, et une compréhension plus profonde du problème a été atteinte. Et au-delà, nous entrevoyons maintenant une belle possibilité de prolonger notre travail pour en généraliser la portée sur le calcul d'invariants en général.

RÉFÉRENCES

S. Altinakar, G. Caporossi, et A. Hertz, “A comparison of integer and constraint programming models for the deficiency problem”, *Computers & Operations Research*, vol. 68, pp. 89–96, 2016.

A. Asratian et C. Casselgren, “On interval edge colorings of (α, β)-biregular bipartite graphs”, *On interval edge colorings of (α, β)-biregular bipartite graphs*, vol. 307, pp. 1951–1956, 2006.

A. Asratian et R. Kamalian, “Interval colorings of the edges of a multigraph”, *Applied Math.*, vol. 5, pp. 25–34, 1987.

M. Bouchard, A. Hertz, et G. Desaulniers, “Lower bounds and a tabu search algorithm for the minimum deficiency problem”, *Journal of Combinatorial Optimization*, vol. 17, 2009.

D. Cohen, P. Jeavons, C. Jefferson, K. Petrie, et B. Smith, “Symmetry definitions for constraint satisfaction problems constraints”, *Kluwer Academic Publishers*, vol. 11, pp. 115–137, 2006.

J. Crawford, M. Ginsberg, E. Luks, et A. Roy, “Symmetry-breaking predicates for search problems”, *KR’96 : Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, California*, pp. 148–159, 1996.

I. Gent, K. Petrie, et J.-F. Puget, “Symmetry in constraint programming”, *Handbook of Constraint Programming, Elsevier*, vol. 2, pp. 329–376, 2006.

K. Giaro, “Interval edge-coloring of graphs”, *Graph colorings, Amercian Mathematical Society*, vol. 352, pp. 105–122, 2004.

——, “The complexity of consecutive δ-coloring of bipartite graphs : 4 is easy, 5 is hard”, *Ars Combinatoria*, vol. 47, pp. 287–298, 1997.

K. Giaro, M. Kubale, et M. Malafiejski, “On the deficiency of bipartite graphs”, *Discrete Applied Mathematics*, vol. 94, pp. 193–203, 1999.

——, “Compact scheduling in open shop with zero-one time operations”, *INFOR*, vol. 37, pp. 37–47, 1999.

——, “Consecutive colorings of the edges of general graphs”, *Discrete Mathematics*, vol. 236, pp. 131–143, 2001.

D. Hanson et C. Loten, “A lower bound for interval colouring of bi-regular bipartite graphs”, *Bulletin of the Institute of Combinatorics and Applications*, vol. 18, pp. 69–74, 1996.

D. Hanson, C. Loten, et B. Toft, “On interval colorings of bi-regular bipartite graphs”, *Ars Combinatoria*, vol. 50, pp. 23–32, 1998.

M. Kubale et A. Nadolski, “Chromatic scheduling in a cyclic open shop”, *European Journal of Operational Research*, vol. 164, pp. 585–591, 2005.

E. Luks et A. Roy, “The complexity of symmetry-breaking formulas”, *Annals of Mathematics and Artificial Intelligence*, vol. 41, pp. 19–45, 2004.

B. McKay, “Practical graph isomorphism”, *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

B. McKay et A. Piperno, “Practical graph isomorphism, ii”, *Preprint submitted to Elsevier, 9 January 2013*, 2013.

——, “Practical graph isomorphism, ii”, *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014.

A. Nadolski, “Compact cyclic edge-colorings of graphs”, *Discrete Mathematics*, vol. 308, pp. 2407–2417, 2008.

J.-F. Pujet, “Breaking symmetries in all different problems”, 2005, pp. 272–277.

A. Pyatkin, “Interval coloring of $(3, 4)$-biregular bipartite graphs having large cubic subgraphs”, *Journal of Graph Theory*, vol. 47, no. 2, pp. 122–128, 2004.

J. Rotman, *Advanced Modern Algebra*, Prentice-Hall, éd., 2002.

A. Schwartz, “The deficiency of a regular graph”, *Discrete Mathematics*, vol. 306, pp. 1947–1954, 2006.

S. Sevastianov, “On interval edge colouring of bipartite graphs”, *Metody Diskretnowo Analiza*, vol. 50, pp. 61–72, 1990.

V. Vizing, “On an estimate of the chromatic class of a p-graph”, *Diskret. Analiz.*, vol. 3, pp. 25–30, 1964.