



**HAL**  
open science

# Problèmes Combinatoires de graphes : de la Fiabilité des réseaux à la coloration de graphes

Corinne Lucet

► **To cite this version:**

Corinne Lucet. Problèmes Combinatoires de graphes : de la Fiabilité des réseaux à la coloration de graphes. Informatique [cs]. Université de Picardie Jules verne, 2019. tel-03234676

**HAL Id: tel-03234676**

**<https://hal.science/tel-03234676>**

Submitted on 25 May 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



UNIVERSITÉ DE PICARDIE JULES VERNE

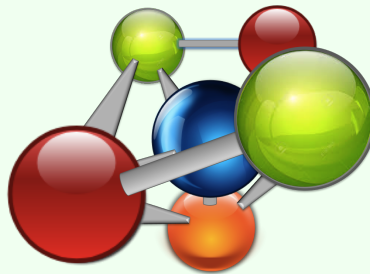
HABILITATION À DIRIGER DES RECHERCHES

CORINNE LUCET

---

**Problèmes Combinatoires de graphes : de la Fiabilité des réseaux à la coloration de graphes**

---



JURY :

Professeur Felip Manyà (rapporteur)  
Professeur Lakhdar Saïs (rapporteur)  
Professeur Gilles Dequen (rapporteur)  
Professeur Jacques Carlier  
Professeur Chumin LI (Tuteur)

Septembre 2019



# Remerciements

Je voudrais tout d'abord remercier Chumin Li, Professeur au laboratoire Modélisation Information et Systèmes, de l'Université de Picardie Jules Verne, pour ses conseils et sa collaboration lors de nos travaux communs, et pour m'avoir encouragée à présenter ce mémoire.

Je suis très honorée que M. Lakhdar Saïs, Professeur au Centre de Recherche en Informatique de Lens, ainsi que M. Felip Manyà, Directeur de Recherche à l'Institut de Recherche en Intelligence Artificielle de Barcelonne, soient membres de mon jury comme rapporteurs.

Merci à M. Gilles Dequen, Professeur et Directeur du laboratoire Modélisation Information et Systèmes d'avoir accepté de rapporter sur ce mémoire. Il a toujours fait preuve d'écoute, de bienveillance, et d'encouragements, qui ne sont pas étrangers à la concrétisation de cette présentation.

Je souhaite témoigner de toute ma reconnaissance à M. Jacques Carlier, aujourd'hui membre de mon jury, et qui fut mon directeur de thèse il y a "quelques" années. Il fut tout au long de ma carrière un soutien et un modèle de sciences et d'humanité.

Je tiens, plus largement, à exprimer ma reconnaissance à toutes celles et à tous ceux qui ont contribué, directement ou indirectement, au bon déroulement de mes travaux. Je souhaite pour cela mettre en avant Laure Brisoux-Devendeville et Yu Li, pour leur amitié et les longues séances de travail, d'échanges et de bonne humeur que nous partageons. Merci également à Juliette Dubois et Céline Joiron pour leur aide et leur amitié au quotidien, et à tous les collègues du laboratoire pour ces agréables moments lors de la "pause café" matinale.

Sur un plan plus personnel, je voudrais témoigner toute ma reconnaissance à ma famille, Pascal, Romane Hugo, Estelle et Antoine, ainsi qu'à mes deux derniers petits bouts Axel et Léa. Cette belle tribu est mon bonheur !





# Corinne LUCET

Cette section présente une synthèse de ma carrière, ainsi que les mots clés associés à ma recherche et mon enseignement. Un curriculum vitae étendu, présentant l'évolution de ma carrière, l'ensemble de mes publications, de mes encadrements et de mes projets se trouve en annexe à la fin de ce document.

## Synthèse de carrière

**1993** Soutenance de ma thèse, Laboratoire HeuDiaSyC, Université de Technologie de Compiègne

**1994** Maître de Conférences Institut Universitaire de Technologie, Département Informatique, Université de Picardie Jules Verne.

Membre du laboratoire HeuDiaSyC, Université de Technologie de Compiègne

**1998** Maître de Conférences Première Classe (ancien système de promotion)

**1999** Membre du laboratoire de recherche en informatique (LaRIA FRE 2733), Université de Picardie Jules Verne, devenu le laboratoire Modélisation, Information et Systèmes (MIS EA 4290) depuis 2008

**2011** Maître de Conférences Hors Classe

**2018** Autorisation à l'inscription en HdR

## Recherche

**Mots-Clés** : Résolution de problèmes NP-difficiles, Graphes, Décomposition arborescente et linéaire, Diagramme de Décision Binaire, Heuristiques et Métaheuristiques, Coloration de graphes, Fiabilité des réseaux, Localisation/Routage, Planification de tâches.

La liste ci-dessous est une sélection de huit articles représentatifs de mes activités de recherche et consultables en annexe du document.

- [1] Jacques Carlier and Corinne Lucet. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics*, 65(1) :141-156, 1996.
- [2] Corinne Lucet, Jean-Francois Manouvrier, and Jacques Carlier. Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs. *Algorithmica*, 27(3) :316-336, 2000.
- [3] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Trans. Reliability*, 56(3) :506-515, 2007.

- [4] Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. *Computers & OR*, 33 :2189-2207, 2006.
- [5] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, and Yu Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663-670, 2010.
- [6] Abdoul Soukour A., Devendeville L., Lucet C., Moukrim A., A Memetic Algorithm for staff scheduling problem in airport security service. *Expert Syst. Appl*, 40(18) :7504-7512, (2013).
- [7] Clément Lecat, Corinne Lucet, and Chu-Min Li. New lower bound for the minimum sum coloring problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4-9, 2017, San Francisco, California, USA., pages 853-859, 2017.
- [8] Clément Lecat, Corinne Lucet, and Chu-Min Li. Minimum sum coloring problem : Upper bounds for the chromatic strength. *Discrete Applied Mathematics*, 233 :71-82, 2017.

## Enseignement

**Mots Clés :** Algorithmique et structures de données, Recherche Opérationnelle, Programmation Linéaire, Théorie des Langages.

Mon service d'enseignement est fait dans son intégralité au département informatique de l'Institut Universitaire de Technologie d'Amiens, au niveau de la 1<sup>ère</sup> et 2<sup>ème</sup> année (L1 et L2). J'interviens également 1<sup>ère</sup> année du cycle ingénieur du CNAM (L3).

**Actuellement :**

- Module M1102 : Initiation à l'algorithmique (L1 - IUT Info)
- Module M1103 : Structures de données (L1 - IUT Info)
- Module M4202C : Recherche Opérationnelle (L2 - IUT Info)
- Module Algorithmique et Programmation (L3 - CNAM)

J'ai enseigné d'autres matières comme la théorie des langages, la programmation dynamique, les bases de données, la programmation linéaire, depuis le début de ma carrière.

Je fus également responsable des stages de 1999 à 2011 au département informatique de l'IUT.

# Table des matières

<b>1</b>	<b>Problème de Fiabilité des réseaux</b>	<b>13</b>
1.1	Introduction . . . . .	13
1.2	Définitions et quelques approches de résolution . . . . .	14
1.2.1	Etats d'un réseau et fiabilité . . . . .	15
1.2.2	Fonctions de communication . . . . .	16
1.2.3	Les méthodes énumératives . . . . .	16
1.2.4	Les réductions séries-parallèles, polygone-à-chaîne et delta-à-étoile . . . . .	17
1.2.5	Factorisation . . . . .	18
1.3	Décomposition Arborescente . . . . .	18
1.3.1	Principe général . . . . .	19
1.3.2	Décomposition arborescente pour la fiabilité des réseaux . . . . .	20
1.3.3	Synthèse des résultats . . . . .	24
1.4	Diagrammes de Décision Binaires . . . . .	24
1.4.1	Définitions des BDD généralités . . . . .	25
1.4.2	Modélisation de la fiabilité des réseaux par les BDD . . . . .	26
1.4.3	Synthèse des Résultats . . . . .	29
1.5	Conclusion . . . . .	29
<b>2</b>	<b>Problèmes de coloration de graphes</b>	<b>33</b>
2.1	Problème du nombre chromatique d'un graphe . . . . .	34
2.1.1	Introduction . . . . .	34
2.1.2	Définition du problème . . . . .	34
2.1.3	Décomposition arborescente pour le problème du nombre chromatique . . . . .	35
2.1.4	Décomposition linéaire pour le problème du nombre chromatique . . . . .	36
2.1.5	Pré-traitements . . . . .	38
2.1.6	Synthèse des résultats sur la coloration . . . . .	38
2.2	Somme chromatique et force d'un graphe . . . . .	39
2.2.1	Introduction . . . . .	39
2.2.2	Définition du problème . . . . .	40
2.2.3	Méthodes approchées . . . . .	41
2.2.4	Formulation SAT . . . . .	47
2.2.5	Synthèse des résultats WP-MaxSAT et WP-MinSAT . . . . .	50
2.2.6	Représentation de l'espace des solutions par Motifs . . . . .	50
2.2.7	Synthèse des résultats Motifs et bornes supérieures de la force . . . . .	54
2.2.8	Synthèse des résultats Motifs et borne inférieure de la somme chromatique : . . . . .	55
2.3	Conclusion . . . . .	56
	<b>Conclusion</b>	<b>57</b>
	<b>Bibliographie</b>	<b>59</b>

<b>A</b>	<b>CURRICULUM VITAE Etendu</b>	<b>65</b>
A.1	Etat Civil . . . . .	65
A.2	Etablissements . . . . .	65
A.3	Titres et Diplômes . . . . .	66
A.4	Synthèse de carrière . . . . .	67
A.5	Activités Scientifiques . . . . .	69
	A.5.1 Présentation des thématiques de recherche . . . . .	69
	A.5.2 Problèmes de Fiabilité des réseaux . . . . .	69
	A.5.3 Problèmes de coloration de graphes . . . . .	72
	A.5.4 Problèmes Logistiques . . . . .	74
A.6	Publications . . . . .	76
	A.6.1 Articles de Livre d'audience internationale . . . . .	76
	A.6.2 Revues d'audience internationale . . . . .	76
	A.6.3 Conférences internationales . . . . .	76
	A.6.4 Conférences nationales . . . . .	78
A.7	Encadrements et animation recherche . . . . .	80
	A.7.1 Encadrement de Thèses . . . . .	80
	A.7.2 Encadrements de Stages de Recherche Master . . . . .	81
	A.7.3 Responsabilités collectives et Organisations . . . . .	83
A.8	Valorisation et projets . . . . .	84
	A.8.1 Projets de recherche . . . . .	84
A.9	Activité pédagogiques . . . . .	86
<b>B</b>	<b>Sélection d'articles</b>	<b>89</b>

# Introduction générale

L'informatique est une science qui a pour vocation de se confronter à des problèmes classiques complexes qu'elle cherche à résoudre, mais elle a également la particularité, de par son développement fulgurant ces dernières décennies d'ouvrir le champ des possibles dans de nombreux domaines et par là-même d'engendrer de nouveaux problèmes.

Ces problèmes nous les rencontrons au quotidien, à travers nos transports, nos communications, nos agendas, la gestion de notre santé, etc. Tous ces domaines sont devenus des systèmes très complexes pour lesquels un grand nombre de données sont à considérer lors de prises de décision qui se doivent d'être de plus en plus rapides dans leur gestion. Parmi ces problèmes certains sont appelés *problèmes combinatoires*, car leur résolution se heurte en général à une explosion du nombre de combinaisons à explorer. Ils appartiennent pour la plupart à la classe des problèmes NP-difficiles, pour lesquels il n'existe pas de méthode efficace en temps pour résoudre leurs instances les plus difficiles.

Mes activités de recherche qui s'effectuent au sein de l'équipe Graphe, Optimisation et Contraintes (GOC) du laboratoire Modélisation Information & Systèmes (MIS/UPJV), se concentrent essentiellement sur l'étude de certains de ces problèmes combinatoires, et plus particulièrement sur ceux qui se modélisent par les objets abstraits que sont les graphes. Les principaux problèmes étudiés sont des problèmes de fiabilité de réseaux, des problèmes de coloration de graphe, comme la détermination du nombre chromatique et la détermination de la somme chromatique, ainsi que des problèmes de pure logistique comme ceux de tournées de véhicules.

Les méthodes d'*optimisation combinatoire* utilisées pour les résoudre, ont pour but d'extraire d'un ensemble fini mais souvent très grand de possibilités la meilleure d'entre elles. Pour cela, elles développent des stratégies exploitant les propriétés structurelles des problèmes qui permettent de concevoir des algorithmes efficaces exacts ou approchés. Lors de mes travaux j'ai pu aborder chacun des problèmes par l'une et l'autre de ces approches.

**Fiabilité des réseaux** Le problème de la *fiabilité des réseaux* est un de ces problèmes NP-difficiles, qui se modélise par un graphe stochastique. Il s'agit de calculer la probabilité qu'un réseau, dont les composants sont susceptibles de défaillances avec certaines probabilités, assure sa fonction de communication à un instant donné. Quelles est la fiabilité en fonction de la structure du réseau ? Comment construire un réseau avec une probabilité de fonctionnement au dessus d'un certain seuil ? Quels sont les composants sensibles dans le réseau ? Voici quelques questions auxquelles nous avons tenté de répondre par le développement de méthodes de décompositions arborescentes du graphe ainsi que d'algorithmes basés sur les puissantes structures que sont les diagrammes de décision binaire. Il s'agit dans ces deux cas d'algorithmes exacts, tirant profit de la structure des graphes modélisant le réseau. Ces travaux, qui furent en partie l'objet de ma thèse, se sont déroulés de 1990 à 2007.

**Nombre chromatique** L'étude et le développement de la méthode de décomposition arborescente de graphe pour la résolution du problème de fiabilité de réseaux, nous a naturellement conduit à aborder un problème phare en optimisation combinatoire, le problème du *nombre chromatique* d'un graphe. La liaison entre ces deux problèmes ne semble pas immédiate, mais dans les faits ils se modélisent de manière similaire pour une résolution par décomposition arborescente. Le problème du nombre chromatique d'un graphe a pour objectif de colorier les sommets du graphe

avec un nombre minimum de couleurs, tout en respectant la contrainte de voisinage qui interdit que deux sommets voisins partagent une même couleur. Ce nombre minimum de couleurs est appelé *nombre chromatique* du graphe. Ce problème permet de modéliser de nombreux problèmes réels comme certains problèmes de planification ou encore des problèmes d’allocations de ressources. Nous avons proposé des algorithmes basés sur une technique de décomposition linéaire du graphe, combinée à des opérateurs de pré-traitements et des calculs de bornes.

**Somme chromatique** A l’issue de ces travaux l’objectif était d’étendre nos algorithmes à la résolution d’un problème de coloration beaucoup moins étudié dans la littérature, celui de la somme chromatique d’un graphe. Il ne s’agit plus de minimiser le nombre de couleurs, mais la somme des poids associés aux couleurs utilisées. Pour ce problème, le nombre de couleurs utilisées dans la solution optimale peut être arbitrairement grand par rapport au nombre chromatique. Ce nombre de couleurs est appelé *la force* du graphe. Cette caractéristique implique que l’ensemble des possibilités à explorer peut être beaucoup plus grand. Ceci nous a conduit à renoncer au développement de méthodes exactes dans un premier temps et à proposer des heuristiques, dont deux algorithmes gloutons MDSAT et MRLF, un algorithme de recherche locale IDCA et un algorithme évolutionnaire MA-MSCP.

La difficulté de développer des méthodes heuristiques robustes et efficaces pour ce problème nous a renvoyé vers la conception d’algorithmes exacts. Premièrement en proposant une formulation MaxSAT et MinSAT afin de bénéficier des puissants solveurs dédiés à ces problèmes. Puis, en développant des algorithmes de branch-and-bound. Le principal défaut de ces méthodes est l’utilisation a priori d’un nombre maximum de couleurs pour définir l’espace des solutions à explorer afin d’y débusquer une solution optimale. Ce handicap nous a poussé à rechercher une borne supérieure de la force de qualité. C’est par la représentation des solutions sous une forme agrégée appelée *motif* que nous avons obtenu d’excellents résultats à la fois sur la borne supérieure de la force et sur la borne inférieure de la somme chromatique.

**Problèmes de logistique** En parallèle de ces deux problèmes plutôt académiques, nous avons également élargi notre domaine d’application en travaillant sous forme de projets de recherche, sur des problèmes de planification d’emplois du temps et des problèmes de transport.

Un de ces projets avait pour thème les problèmes de transports et plus spécifiquement le problème de localisation/routage. Ce problème provient de l’idée de combiner deux niveaux de décision de logistique : la localisation de dépôts où sont stockées les ressources et l’élaboration de tournées de véhicules afin de distribuer ces ressources et satisfaire les demandes. Les travaux effectués sur ce problème le furent dans le cadre du projet *PRIMA* sur la résolution de problèmes de localisation et routage en milieu urbain.

Les autres projets sont tous des problèmes de planification d’emploi du temps. Le projet *PLAN-AIR*, avait pour sujet la planification des personnels dans la sûreté aérienne. Il s’agissait d’une collaboration entre l’entreprise ICTS France, le laboratoire MIS/UPJV et le laboratoire HeuDiaSyC/UTC. *SimuStorE* est un projet en cours, sur la planification pour un centre de pédagogie active, spécialisé en santé, mettant en relation le laboratoire MIS/UPJV et le centre de pédagogie active SimuSanté/CHU-Amiens. Sa particularité est la grande diversité des types de ressources. Différents acteurs de la santé sont concernés par ce centre de formation, qui propose des équipements (salles d’opération, salles de cours, pharmacie, salles vidéo, etc.) très variés, permettant un apprentissage par la simulation. Enfin, depuis Octobre 2017, le projet *LORH* rassemble le laboratoire MIS et l’entreprise EVOLUCARE sur un problème d’optimisation du parcours patient en milieu hospitalier. L’objectif est de proposer un outil de planification répondant à un grand nombre de scénarios de la vie hospitalière.

Pour l’ensemble de ces problèmes de logistique, nous développons des méthodes heuristiques basées sur des concepts d’algorithmes évolutionnaires. Cette partie de mes travaux ne sera pas développée dans ce document, mais l’ensemble des articles publiés sur ces thématiques ainsi qu’une présentation

succincte des sujets et des méthodes développées sont listés dans le Curriculum Vitae étendu (Annexe A).

Ce document est composé de deux principaux chapitres. Le chapitre 1 présente les principales notions, méthodes et résultats associés aux différents problèmes de fiabilité des réseaux. Le chapitre 2 donne les définitions des deux problèmes de coloration, le nombre chromatique et la somme chromatique. Pour chacun, il synthétise les méthodes développées, et les résultats obtenus. Enfin, nous présentons une conclusion et la mise en perspective de ces travaux.

L'annexe A est un CV étendu dans lequel se trouve une synthèse de ma carrière, ma liste de publications, d'encadrements et de mes projets. A la suite de cette annexe vous trouverez une sélection de mes articles.





# Chapitre 1

## Problème de Fiabilité des réseaux

### Sommaire

---

<b>1.1</b>	<b>Introduction</b>	<b>13</b>
<b>1.2</b>	<b>Définitions et quelques approches de résolution</b>	<b>14</b>
1.2.1	Etats d'un réseau et fiabilité	15
1.2.2	Fonctions de communication	16
1.2.3	Les méthodes énumératives	16
1.2.4	Les réductions séries-parallèles, polygone-à-chaîne et delta-à-étoile	17
1.2.5	Factorisation	18
<b>1.3</b>	<b>Décomposition Arborescente</b>	<b>18</b>
1.3.1	Principe général	19
1.3.2	Décomposition arborescente pour la fiabilité des réseaux	20
1.3.3	Synthèse des résultats	24
<b>1.4</b>	<b>Diagrammes de Décision Binaires</b>	<b>24</b>
1.4.1	Définitions des BDD généralités	25
1.4.2	Modélisation de la fiabilité des réseaux par les BDD	26
1.4.3	Synthèse des Résultats	29
<b>1.5</b>	<b>Conclusion</b>	<b>29</b>

---

### 1.1 Introduction

La sûreté de fonctionnement est un objectif industriel de grande importance qui doit être prise en compte dès la conception des systèmes, et administrée jusqu'à leur démantèlement. Les procédés de fabrication intègrent généralement des processus de certification dont la fiabilité prévisionnelle du système est souvent un des points obligés. En terme de systèmes complexes, les réseaux occupent une place de choix, tant leur taille et leur complexité se sont développées depuis le milieu du siècle dernier. Leur conception, leur validation et leur maintenance est évidemment l'objet du travail de nombreux chercheurs. Les problèmes étudiés sont très variés, ils tendent à classifier les réseaux suivant des propriétés structurelles et topologiques, à dimensionner leur capacité, à organiser les flux qui les traversent mais aussi à déterminer leur *fiabilité*. La fiabilité au sens large, est définie comme "la probabilité de fonctionnement sans défaillance d'un dispositif, dans des conditions déterminées et pour une période de temps définie". Cette métrique représente une mesure de la qualité de service que pourra assurer un réseau donné, en fonction de sa structure et de la nature de ses composants.

Les réseaux sur lesquels nous avons travaillé sont des réseaux de communication. Pour ceux-ci, le problème de la fiabilité consiste alors à calculer la probabilité que sa fonction de communication entre différents sites soit assurée, sachant que ses composants, les sites et/ou les liaisons, sont sujets à des défaillances de manière indépendante. Il a été montré que le problème du calcul de la fiabilité de réseaux est NP-difficile [9, 10]. Il n'existe donc pas d'algorithmes efficaces permettant de calculer

la fiabilité d'un réseau dans le cas général, néanmoins, pour certaines familles de graphes telle que celle regroupant les graphes séries-parallèles, des algorithmes polynomiaux peuvent en calculer la fiabilité.

Les méthodes développées et présentées dans la littérature se divisent en deux catégories : les méthodes exactes qui fournissent la valeur exacte de la probabilité de fonctionnement du réseau, et les méthodes approchées qui en donnent une borne inférieure ou supérieure. Lors des travaux que nous avons mener sur ces problèmes, nous nous sommes principalement intéressés à deux types de méthodes exactes. La décomposition arborescente et les diagrammes de décision binaires.

L'étude de la méthode de décomposition arborescente d'un graphe pour résoudre un problème de fiabilité de reseaux fut l'objet de ma thèse de 1990 à 1993. Ces travaux ont été menés sous la direction du Professeur Jacques Carlier (Heudiasyc/UTC). Nous avons ensuite prolongé notre collaboration sur ce sujet à travers le co-encadrement de la thèse de M. Jean-François Manouvrier (doctorant UTC) de 1995 à 1998. Notre étude avait pour but d'étendre notre expertise de la décomposition pour résoudre des problèmes de fiabilité pour des fonctions de communication étendues, ainsi que son application à d'autres problèmes de graphes, comme celui du voyageur de commerce ou des arbres minimaux de Steiner. L'ensemble de ces travaux ont donné lieu à des publications dans des revues internationales comme *Discrete Applied Mathematics* [1] et *ALGORITHMICA* [2] ainsi que dans des conférences internationales comme *Graphs and Optimization (GO I 1992)*, *European Chapter on Combinatorial Optimization, (ECCO VI 1993 et VII 1994)*, *Conférence Francophone de Recherche Opérationnelle (FRANCORO 1995)*, et *Conference on Principles of Distributed Systems (OPODIS 1997)*.

La seconde méthode que nous avons étudiée pour le calcul de la fiabilité des réseaux, était basée sur les *diagrammes de décision binaires* en collaboration avec le Professeur Nikolaos Limnios du Laboratoire de Mathématiques Appliquées de Compiègne (LMAC/UTC). Cette collaboration eu lieu autour du co-encadrement de la thèse de M. Gary Hardy (doctorant UPJV), financée par un projet du Conseil Régional de Picardie de 2004 à 2007. Ces travaux ont été publiés comme chapitre de livre dans *Recent Advancement of Stochastic Operations Research, World Scientific (2007)*[11], dans des revues internationales *IEEE Trans. on Reliability* [3], *SIGMETRICS Performance Evaluation Review* [12] et des conférences internationales comme par exemple *International Symposium on Computer Performance, Modeling, Measurements and Evaluation, (PERFORMANCE 2005)* ou *International Symposium on Applied Stochastic Models and Data Analysis (ASMDA 2005)*. Gary Hardy a également présenté ses travaux à deux reprises en 2005 et 2006 au congrès national de la ROADEF.

Ce chapitre présente premièrement, les principales définitions relatives au problème de la fiabilité des réseaux, les différentes fonctions de communication que nous avons étudiées et les grands principes des méthodes de la littérature. Ensuite, nous présentons la méthode de décomposition arborescente comme méthode générale, puis telle que nous l'avons appliquée au problème de fiabilité des réseaux. Enfin nous présentons les puissantes structures que sont les Diagrammes de Décision Binaires et leur utilisation pour résoudre le problème de fiabilité des réseaux et un problème d'optimisation de conception de réseaux. Nous terminons ce chapitre par une conclusion sur l'ensemble de ces travaux.

## 1.2 Définitions et quelques approches de résolution

Dans un problème de calcul de fiabilité, le réseau de communication est modélisé par un graphe stochastique,  $G = (V, E, p)$ , chaque site du réseau est représenté par un sommet  $v \in V$  du graphe, et chaque lien entre deux sites par une arête  $e = (v_1, v_2) \in E$ . Une probabilité de fonctionnement  $p_i$  est associée à chaque composant  $i$  du graphe. Cette valeur comprise entre 0 et 1 représente la probabilité que le composant soit en état de fonctionnement à un instant donné. Ces données sont supposées constantes sur l'intervalle de temps considéré, et indépendantes les unes des autres. Les problèmes de fiabilité de réseaux se conjuguent en fonction de la nature des composants. Les

sommets et les arêtes peuvent être considérés comme parfaits ou susceptibles de défaillance. La fonction de communication est également un facteur changeant. Par exemple la fonction *connectivité tous-terminaux* exigera que tous les sites du réseau soient reliés, c'est à dire qu'il existe un chemin de composants en état de fonctionnement reliant tout couple de sites. Alors que la *connectivité K-terminaux* demande que tous les sites d'un sous ensemble donné  $K \subset V$  soient reliés deux à deux. La *fiabilité 2-terminaux* est un cas particulier de cette dernière, seuls deux sites identifiés  $s$  et  $t$  doivent être reliés par au moins une chaîne en état de fonctionnement. La figure 1.1, dans laquelle les sommets noirs représentent les sommets terminaux, illustre ces trois fonctions de communication.

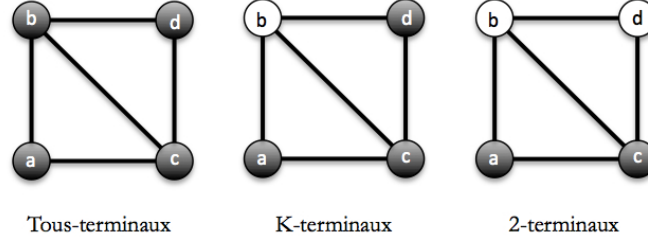


FIGURE 1.1 – Fonctions de communication

Dans la suite de ce chapitre, pour plus de clareté, nous considérerons que les sommets sont parfaits, donc que la probabilité associée à chaque sommet est égale à 1. On notera  $n$  la cardinalité de  $V$  et  $m$  la cardinalité de  $E$ .

### 1.2.1 Etats d'un réseau et fiabilité

**Définition 1 (Etat du graphe)** Un état  $\varphi_k$  d'un graphe  $G = (V, E, p)$  est représenté par un vecteur  $(x_1, x_2, \dots, x_m)$  où la variable booléenne  $x_i$  vaut 1 si l'arête  $e_i$  fonctionne et 0 si celle-ci est défaillante.  $(x_1, x_2, \dots, x_m)$  est appelé vecteur d'état du graphe.

Chaque arête possédant deux états (fonctionnement/défaillance), le nombre d'états associés au graphe  $G$  est  $2^m$ . La probabilité  $p_e$  représente la probabilité que l'arête  $e$  soit en état de fonctionnement à l'instant  $t$  et la probabilité associée à un état donné  $\varphi_k$  est alors définie par l'équation 1.1.

$$P(\varphi_k) = \prod_{e \in E} [x_e \cdot p_e + (1 - x_e)(1 - p_e)] \quad (1.1)$$

A chaque état  $\varphi_k$  on associe un graphe partiel  $G(\varphi_k)$  de  $G$ , constitué des arêtes en état de fonctionnement dans  $\varphi_k$ .  $G(\varphi_k) = (V, E')$  tel que  $E' \subseteq E$  et  $e \in E'$  ssi  $e \in E$  et  $x_e = 1$ .

La figure 1.2 montre deux graphes partiels associés au graphe  $G$ .  $G(\varphi_1)$  est le graphe partiel engendré par  $\varphi_1 = (1, 0, 0, 1, 1)$  et  $G(\varphi_2)$  est le graphe partiel engendré par  $\varphi_2 = (1, 1, 0, 0, 0)$ .

**Définition 2 (Etat de fonctionnement)** Un état  $\varphi_k$  est appelé état de fonctionnement, si la fonction de communication est assurée dans le graphe  $G(\varphi_k)$  et état de défaillance sinon. On note  $\phi = \{\varphi_1, \varphi_2, \dots, \varphi_{2^m}\}$  l'ensemble des  $2^m$  états possibles du graphe  $G$ .  $\phi$  est décomposé en deux sous-ensembles,  $\phi_1$  l'ensemble des états de fonctionnement et  $\phi_0$  l'ensemble des états de défaillance.

**Définition 3 (Fiabilité du réseau)** La fiabilité  $R(G)$ , du réseau représenté par le graphe  $G$ , est alors calculée par l'équation 1.2.

$$R(G) = \sum_{\varphi_k \in \phi_1} P(\varphi_k) \quad (1.2)$$

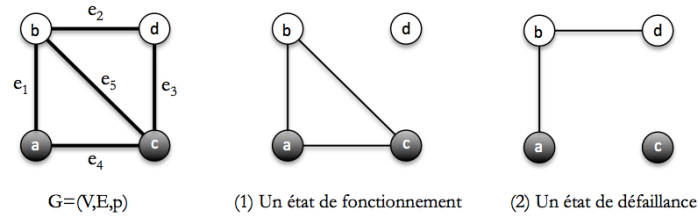


FIGURE 1.2 – (1)  $G(\varphi_1)$   $\varphi_1$  état de fonctionnement ; (2)  $G(\varphi_2)$   $\varphi_2$  état de défaillance.

### 1.2.2 Fonctions de communication

Afin de définir plus précisément les fonctions de communication qui décident de la qualification de "fonctionnement" ou de "défaillance" pour un état du réseau, nous introduisons dans la définition du problème la notion de *sommets terminaux*, et nous noterons  $K$  l'ensemble de ces sommets ( $K \subseteq V$ ).

**Définition 4 (Fonction de communication)** *La fonction de communication est assurée dans le graphe  $G(\varphi_k)$  si et seulement si il existe une chaîne entre tout couple de sommets terminaux de l'ensemble  $K$ .*

La fiabilité du réseau est donc la probabilité que tous les sommets de  $K$  soient reliés à un instant  $t$ .

- Si  $K = V$  la fiabilité du réseau est appelée **fiabilité tous-terminaux**.
- Si  $K = \{s, t\}$   $s, t \in V$  la fiabilité du réseau est appelée **fiabilité 2-terminaux ou  $\{s, t\}$ -terminaux**.
- Si  $2 < |K| < |V|$  la fiabilité du réseau est appelée **fiabilité  $K$ -terminaux**.

Dans l'exemple de la figure 1.2, si on considère que  $K = \{a, c\}$ ,  $\varphi_1$  est un état de fonctionnement et  $\varphi_2$  est un état de défaillance.

### 1.2.3 Les méthodes énumératives

Les premières méthodes de résolution proposées dans la littérature sont des méthodes basées sur l'énumération d'états, et donc sur le développement d'un arbre d'états comme illustré sur la figure 1.3. L'évaluation probabiliste pour chacun de ces états est très simple (cf. équation 1.2). Des stratégies de réduction de cet arbre ont été proposées dans la littérature [13], généralement basées sur le choix dynamique de l'arête pour chaque branchement dans la génération de l'arbre d'états. Ce genre de technique permet de diminuer la taille de l'arbre, mais l'évaluation de la fiabilité reste exponentielle et difficile à calculer pour des réseaux de taille modeste.

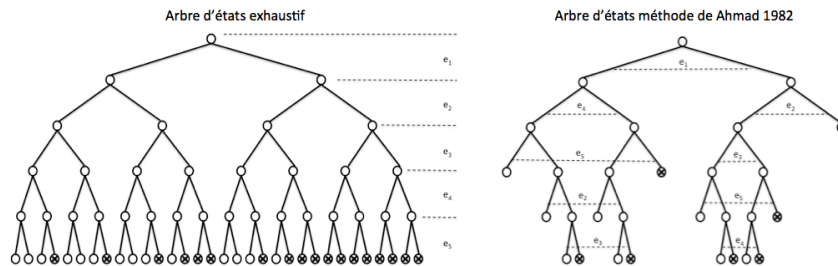


FIGURE 1.3 – Comparaison de la taille des arbres d'états correspondant au graphe de la figure 1.2

Une approche similaire consiste à énumérer les chemins minimaux ou les coupes minimales du graphe. Un chemin minimal est un ensemble minimal de composants (d'arêtes) du graphe permettant de relier les sommets terminaux. Ainsi dans le cas de la fiabilité 2-terminaux, les chemins minimaux sont les chemins élémentaires reliant le sommet  $s$  au sommet  $t$ . Pour la fiabilité tous-terminaux il s'agit des arbres couvrants et pour la fiabilité K-terminaux des arbres de Steiner. Le fonctionnement des chemins assure le fonctionnement du réseau.

Les coupes minimales sont des sous ensembles d'arêtes tels que la défaillance des éléments de la coupe entraîne la défaillance du réseau. Malheureusement ces chemins minimaux, comme ces coupes minimales, ne sont généralement pas disjoints. L'évaluation probabiliste nécessite de coûteux et délicats calculs mettant en œuvre la somme des produits disjoints ou la formule d'inclusion-exclusion de Poincaré. On trouve dans la littérature, dès le début des années 1980, de très nombreux algorithmes permettant d'énumérer ces chemins et ces coupes [14],[15], [16]. Mais le traitement de cas réels utilise très rarement ce type d'approches, car leur complexité est exponentielle en fonction de la taille du réseaux, et ne traitent donc que des instances de taille très modeste.

### 1.2.4 Les réductions séries- parallèles, polygone-à-chaîne et delta-à-étoile

Les opérateurs de réduction consistent à réduire le graphe en nombre de sommets et d'arêtes tout en maintenant l'information nécessaire au calcul de la fiabilité. Les principaux travaux relatifs aux réductions sont dus à Satyanarayana et Wood [17, 18, 19, 20] et Rosenthal [21, 22] qui dans les années 1980 ont proposés un ensemble de transformations du graphe, répertoriées sous les noms de *réduction série-parallèle*, *réduction polygone-à-chaîne* et *réduction delta-à-étoile*. Le principe d'une réduction consiste à remplacer une sous-structure du graphe  $G$  par une autre. On obtient alors un nouveau graphe  $G'$  tel que  $R(G) = \delta R(G')$ .  $\delta$  est un coefficient dépendant de la réduction appliquée et les probabilités de fonctionnement des éléments remplaçants sont calculées en fonction des probabilités des éléments remplacés.

Quelques exemples de ces réductions sont illustrés par les figures 1.4, 1.5, et 1.6. Nous noterons par exemple que les réductions séries- parallèles permettent de calculer la fiabilité de certains réseaux en temps polynomial. Ces réseaux sont ceux représentés par un *graphe série-parallèle*, qui se réduit à un arbre par applications successives des réductions séries- parallèles.

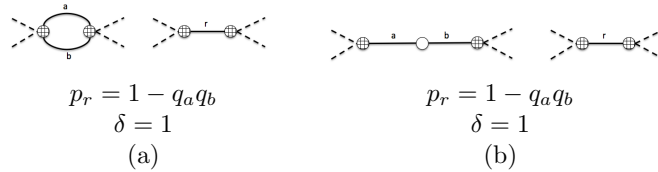


FIGURE 1.4 – (a) Réduction parallèle - (b) Réduction série

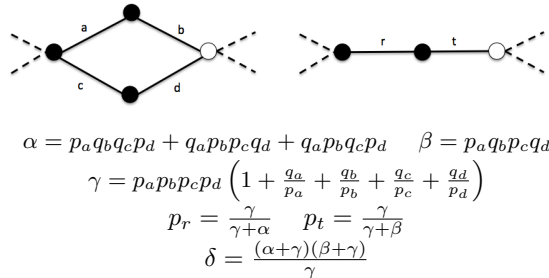


FIGURE 1.5 – Réduction Polygone-à-chaîne - K est réduit d'un sommet.

Ces réductions peuvent être appliquées de manière itérative. Nous noterons ainsi que l'application d'une réduction delta-à-étoile permet de diminuer les degrés des sommets impliqués et ainsi

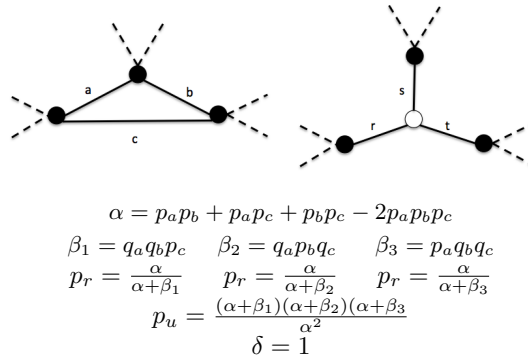


FIGURE 1.6 – Réduction Delta-à-étoile - Un sommet de plus mais les degrés ont diminués.

rendre possibles des réductions séries-parallèles ou polygone-à-chaine.

### 1.2.5 Factorisation

La factorisation est une interprétation topologique de l'espace d'états et s'applique à n'importe quel système binaire, composé des éléments  $e_1, e_2, \dots, e_m$  dont les probabilités de fonctionnement associées sont  $p_1, p_2, \dots, p_m$ . La factorisation s'appuie sur la décomposition de Shannon, qui s'effectue relativement à un de ces composants en fixant pour celui-ci son état. L'espace d'état est alors divisé en deux sous-espaces, un pour lequel le composant est considéré comme défaillant et l'autre pour lequel ce même composant est considéré comme fonctionnant. Dans le cas de la fiabilité K-terminaux, avec sommets parfaits, la fiabilité du graphe  $G(V, E, K, p)$  pour la factorisation de l'arête  $e_i$  s'exprime suivant l'équation 1.3.

$$R(G) = p_i \cdot R(G * e_i) + (1 - p_i) \cdot R(G \setminus e_i) \quad (1.3)$$

$G * e_i$  est le graphe obtenu par contraction de l'arête  $e_i$  (fusion des sommets incidents à  $e_i$ ) et  $G \setminus e_i$  est le graphe obtenu par suppression de  $e_i$ . La factorisation est généralement combinée aux réductions pour calculer la fiabilité. En effet, lorsqu'il n'est plus possible d'appliquer des réductions, la factorisation d'une arête "bien choisie" peut permettre de nouvelles réductions. Ces techniques ont été généralisées aux problèmes de fiabilité des réseaux avec sommets défaillants par Théologou et Carlier [23].

## 1.3 Décomposition Arborescente

La notion de décomposition arborescente est issue de la communauté de chercheurs en théorie des graphes, plus particulièrement des travaux de Robertson et Seymour dans les années 1980 [24, 25]. Hans Bodlaender a présenté dans la littérature un grand nombre de résultats sur la décomposition arborescente des graphes [26, 27, 28]. Cette notion est utilisée à la fois dans la classification des graphes en familles partageant certaines caractéristiques, en théorie des grammaires de graphes, et pour la partie qui nous concerne le plus, en construction d'algorithmes polynomiaux pour résoudre des problèmes NP-complets. En effet, pour certains de ces problèmes, et pour des familles de graphes spécifiques, il est possible de construire des algorithmes de complexité polynomiale en fonction de la taille de l'instance. On a vu la section 1.2.4 que la fiabilité des graphes séries-parallèles est calculable en temps polynomial. Cette classe de graphes "séries-parallèles" est en directe relation avec la notion de décomposition arborescente et de la caractéristique de "largeur arborescente" que l'on peut associer à chaque graphe.

### 1.3.1 Principe général

La méthode de décomposition arborescente (ou décomposition-arbre) d'un graphe est une méthode généraliste, permettant de résoudre de manière exacte certains problèmes NP-complets pour des familles de graphes spécifiques, qui sont appelées les "k-arbres partiels". Elle consiste à partitionner le graphe en sous-ensembles de sommets, de telle sorte que la structure du problème soit préservée. Les sous-ensembles de sommets (parties) ainsi obtenus sont reliés les uns aux autres relativement aux sommets qu'ils partagent, pour former un "arbre de parties", d'où la dénomination de *décomposition arborescente*. Dans la définition 5 d'une décomposition arborescente d'un graphe  $G = (V, E)$ , les nœuds  $N_i$  de l'arbre sont des sous-ensembles de sommets, et l'union des nœuds  $N_i$  couvre  $V$ .

**Définition 5 (Décomposition arborescente)** Soit  $G = (V, E)$  un graphe, une décomposition arborescente de  $G$  est un couple  $\mathcal{D}_t = (T = (N, A); f)$  où  $T$  est un arbre dont les nœuds  $N_i \in N$  sont des sous-ensembles de sommets de  $V$ . On associe à  $\mathcal{D}_t$  une fonction  $f$ , qui à tout sous ensemble de sommets  $N_i$  fait correspondre un sous graphe  $G(N_i) = (N_i, E_i)$  tel que si  $(u, v) \in E_i$  alors  $u \in N_i$  et  $v \in N_i$ . La décomposition arborescente  $\mathcal{D}_t$  satisfait les conditions suivantes :

- $\forall (u, v) \in E, \exists N_i \in N$  tel que  $u \in N_i$  et  $v \in N_i$
- Les graphes  $G(N_i)$  sont disjoints en arête :  $\forall i \neq j, E_i \cap E_j = \emptyset$
- $\bigcup N_i = V$
- Pour tout sommet  $v \in V$ , l'ensemble des nœuds  $N_i$  tels que  $v \in N_i$  induit un sous arbre connexe de  $T$ .

**Définition 6 (Largeur d'une décomposition)** Soit  $\mathcal{D}_t = (T = (N, A); f)$  une décomposition arborescente du graphe  $G$ , on appelle **largeur de  $\mathcal{D}_t$**  la grandeur suivante :

$$twd(\mathcal{D}_t) = \text{Max}\{|N_i|; N_i \in N\}$$

**Définition 7 (Largeur arborescente)** On appelle **largeur arborescente d'un graphe  $G$** , la grandeur suivante :

$$twd(G) = \text{Min}\{twd(\mathcal{D}_t); \mathcal{D}_t \text{ est une décomposition arborescente de } G\}$$

La figure 1.8 montre une décomposition arborescente  $\mathcal{D}_t$  d'un graphe  $G$ , composé d'un arbre de 6 nœuds et dont la largeur correspond à la cardinalité du nœud  $N_6$  qui vaut 4. On remarquera que les sous-graphes induits par la fonction  $f$  sont disjoints en arêtes.

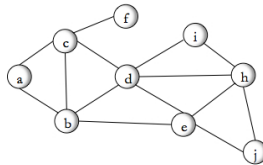
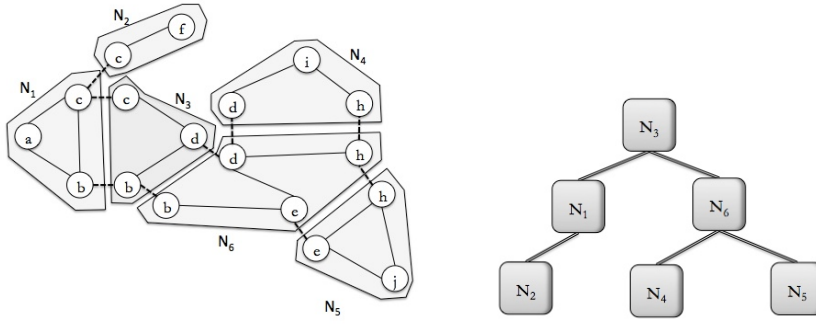


FIGURE 1.7 – Un graphe  $G$

**Résolution du problème** La résolution d'un problème  $\mathcal{P}$  sur le graphe  $G$  se calcule alors à partir des solutions partielles du problème pour les différents sous-graphes  $G(N_i)$ . Premièrement, les solutions partielles des feuilles sont générées et évaluées, puis pour chaque nœud interne  $N_k$  de  $T$ , ses solutions partielles sont également générées, et évaluées en les combinant aux solutions partielles de ses fils  $N_i$  et  $N_j$ . Ainsi, chaque solution partielle du nœud interne  $N_k$  représente en réalité une solution partielle du sous-graphe  $G(T(N_k))$ , induit par l'ensemble des sommets impliqués dans le sous-arbre de  $T$  de racine  $N_k$ .

Sur l'exemple de la figure 1.8, la résolution d'un problème  $\mathcal{P}$  consistera à générer toutes les solutions partielles de  $N_4$  et de  $N_5$ , puis de les combiner afin d'évaluer les solutions partielles de  $N_6$ . Ensuite seront générées les solutions partielles de  $N_2$  qui permettront de générer et évaluer les solutions partielles de  $N_1$ . Enfin, le problème  $\mathcal{P}$  sera résolu par la combinaison des solutions de  $N_6$  et  $N_1$  afin d'évaluer les solutions de la racine  $N_3$ .



FIGURE 1.8 – Décomposition arborescente  $\mathcal{D}_t = (N, A)$  associé au graphe de la figure 1.7

**Complexité** Pour un problème NP-complets  $\mathcal{P}$ , la complexité temporelle d'une méthode énumérative est généralement une fonction exponentielle de la taille du graphe, c'est-à-dire, son nombre de sommets (et/ou son nombre d'arêtes), que l'on notera  $O(e_n)$ . Or, si on considère une décomposition arborescente  $\mathcal{D}_t$  du graphe, composée de  $\alpha$  nœuds  $N_i$ . Chaque résolution du problème  $\mathcal{P}$  sur un nœud  $N_i$  sera de complexité  $O(e_{|N_i|})$ . Par conséquent, la résolution du problème  $\mathcal{P}$  par une décomposition arborescente sera de complexité  $O(\alpha \times e_{wd(\mathcal{D}_t)})$  et donc de manière plus générale en  $O(n \times e_{twid(G)})$ . Dans le cas de familles spécifiques comme celle des  $k$ -arbres partiels, la largeur arborescente est bornée par  $k$ . Il est donc possible lorsque cette valeur  $k$  est petite de fournir une solution optimale en un temps polynomial en fonction de la taille du graphe. Un  $k$ -arbre partiel est un graphe partiel d'un graphe de la famille des  $k$ -arbres, tel que le décrit la définition 8.

**Définition 8 (k-arbre)** Soit un graphe  $G = (V, E)$ .  $G$  est un **k-arbre** si une des deux propositions suivantes est vérifiée :

- (i)  $G$  est un graphe complet de  $k$  sommets
- (ii)  $G$  a un sommet  $v$  de degré  $k$ , tel que le sous-graphe induit par le voisinage de  $v$  est une clique et  $G \setminus \{v\}$  est un  $k$ -arbre.

Tous les problèmes de graphes ne peuvent être résolus par une méthode de décomposition arborescente. De nombreux auteurs, comme Courcelle [29], Boadlander [30] et Arnborg [31] ont tenté de classifier les problèmes de graphes pour identifier ceux potentiellement solvables par cette méthode. Ils ont établis par exemple que tous les problèmes formulables en logique du second ordre monadique sont des problèmes à états finis et que tous les problèmes à états finis peuvent être résolus par une décomposition arborescente. Ils ont également montré que la classe des graphes  $k$ -décomposables, pour lesquels il existe une décomposition de largeur  $k$ , est exactement la classe des  $k$ -arbres partiels. Ainsi le graphe de la figure 1.8 est un 2-arbre partiel aussi appelé graphe série-parallèle. La largeur arborescente des graphes de cette famille est égale à 2. La décomposition proposée sur la figure 1.8 n'est donc pas optimale.

Mais lorsqu'un problème se pose, il ne suffit pas de constater qu'il appartient à cette classe, faut-il encore déterminer une décomposition arborescente de largeur-arbre  $k$  bornée. Or, pour un graphe  $G$  donné, et une valeur  $k$  donnée, répondre à la question " la largeur-arbre de  $G$  est-elle inférieure ou égale à  $k$ " est un problème NP-Complet [32].

### 1.3.2 Décomposition arborescente pour la fiabilité des réseaux

Pour le problème de la fiabilité des réseaux, bien que le principe de la méthode de décomposition arborescente du graphe stochastique fut énoncé par Rosenthal en 1977, aucune expérimentation permettant de mettre en évidence l'efficacité de cette technique sur des réseaux réels n'existait lors des débuts de nos travaux. Nous nous sommes donc focalisés sur l'élaboration et l'implémentation d'algorithmes basés sur la décomposition arborescente pour la résolution de ce problème.

**Solutions Partielles** Pour chaque problème il est nécessaire de définir les solutions partielles associées aux nœuds de la décomposition. Pour résoudre le problème de fiabilité des réseaux, les solutions partielles de chaque nœud  $N_i$  représentent les différents états de connexion des sommets, c'est à dire leur organisation en composantes connexes et leur connexion au sommets terminaux. Pour simplifier nous considèrerons le problème *tous terminaux*. Pour celui-ci, les solutions partielles sont modélisées par les partitions des sommets des  $N_i$ . Si nous considérons le nœud  $N_1 = \{a, b, c\}$  de la figure 1.8, alors l'ensemble des solutions partielles associées est le suivant :

- $C_1^i = [a][b][c]$  :  $a, b$  et  $c$  sont tous dans des composantes connexes différentes.
- $C_2^i = [ab][c]$  :  $a$  et  $b$  sont dans une même composante alors que  $c$  est dans une composante connexe différente.
- $C_3^i = [a][bc]$  :  $c$  et  $b$  sont dans une même composante alors que  $a$  est dans une composante connexe différente.
- $C_4^i = [ac][b]$  :  $a$  et  $c$  sont dans une même composante alors que  $b$  est dans une composante connexe différente.
- $C_5^i = [abc]$  :  $a, b$  et  $c$  sont tous dans une même composante connexe.

La complexité de résolution pour chacun des nœuds  $N_i$  est donc relative au nombre de partitions de l'ensemble  $N_i$ , ce que représente les nombres de stirling de seconde espèce. Ainsi, si on note  $|N_i| = n_i$  et  $\Psi(N_i)$  l'ensemble de ses partitions, le nombre de solutions partielles est donné par l'équation 1.4 et son approximation proposée par Hardy et Ramanujan [33] par l'équation 1.5). La complexité de la résolution de la fiabilité d'un réseau représenté par un graphe  $G = (V, E, p)$  est donc en  $\mathcal{O}(n.e^{n_{max}})$ , avec  $n_{max} = \max\{|N_i|, \forall N_i \in T\}$

$$A(n_i, k) = A(n_i - 1, k - 1) * k + A(n_i - 1, k) \text{ pour } 2 \leq n_i; 2 \leq k \leq n_i$$

$$A(n_i, 1) = 1 \quad \forall n_i \geq 1$$

$$\Psi(N_i) = \sum_{k=1}^{n_i} A(n_i, k) \quad (1.4)$$

$$|\Psi(N_i)| \approx \frac{1}{4n_i\sqrt{3}} e^{\pi\sqrt{2n_i/3}} \quad (1.5)$$

**Evaluation des solutions partielles aux feuilles de  $\mathcal{D}_t$**  A chaque solution partielle  $C_j^i$  est associée la probabilité  $P(C_j^i)$  d'avoir l'état de connexion des sommets qu'elle représente. Le tableau de la figure 1.9 donne pour chaque solution partielle du nœud  $N_5 = \{e, h, j\}$  de la décomposition de la figure 1.8, la probabilité associée, qui est déterminée en fonction des probabilité de fonctionnement associées aux arêtes.

Solution Partielle	Probabilité associée
$C_1^5 = [e][h][j]$	$(1 - p_{eh})(1 - p_{ej})(1 - p_{hj})$
$C_2^5 = [e h][j]$	$p_{eh}(1 - p_{ej})(1 - p_{hj})$
$C_3^5 = [e j][h]$	$(1 - p_{eh})p_{ej}(1 - p_{hj})$
$C_4^5 = [e][j h]$	$(1 - p_{eh})(1 - p_{ej})p_{hj}$
$C_5^5 = [e h j]$	$(1 - p_{eh})p_{ej}p_{hj} + p_{eh}(1 - p_{ej})p_{hj} + p_{eh}p_{ej}(1 - p_{hj}) + p_{eh}p_{ej}p_{hj}$

FIGURE 1.9 – Evaluation des solutions partielles de  $N_5$  de la figure 1.8

**Evaluation des solutions partielles pour un nœud interne de  $\mathcal{D}_t$**  Considérons un nœud interne  $N_k$  d'une décomposition arborescente  $\mathcal{D}_t$ , ayant deux nœuds fils  $N_i$  et  $N_j$ . Soient  $C_x^i = B_1, B_2, \dots, B_r$  une partition de l'ensemble  $N_i$ . On définit l'intersection entre  $C_x^i$  et  $N_k$  comme

étant égale à la partition  $C_x^i \cap N_k = B_1 \cap N_k, B_2 \cap N_k, \dots, B_r \cap N_k$ . La fusion de deux blocs  $B_i$  et  $B_j$  d'une partition forme un seul bloc  $B_i \cup B_j$ .

**Définition 9 (Solutions partielles compatibles)** *La solution partielle  $C_x^i = B_1, B_2, \dots, B_r$  de  $N_i$  et la solution partielle  $C_y^k = B'_1, B'_2, \dots, B'_l$  de  $N_k$  sont compatibles si elles vérifient les propriétés suivantes :*

- $\forall \alpha = 1, \dots, r, B_\alpha \cap N_k \neq \emptyset$
- *Pour tout bloc  $B'_y \cap N_i$  de  $C_y^k \cap N_i$  il existe un ensemble de bloc(s) de  $B_x \cap N_k$  dont la fusion coïncide exactement à  $B'_y \cap N_i$ .*

Alors, l'évaluation de la solution partielle  $C_y^k$  de  $N_k$  est donnée par l'équation 1.6.

$$P(C_y^k) = P'(C_y^k) \times \sum_{C_x^i \text{ compatible } C_y^k} \left( P(C_x^i) \times \sum_{C_z^j \text{ compatible } C_y^k} P(C_z^j) \right) \quad (1.6)$$

$P'(C_y^k)$  représente l'évaluation de la partition  $C_y^k$  si  $N_k$  était une feuille.

Sur l'exemple de la figure 1.8, la partition [a b c] de  $N_1$  est compatible avec les partitions [b c d], [b d][c] et [b][c d] de  $N_3$  et avec la partition [c f] de  $N_2$ .

La difficulté de trouver une décomposition arborescente et la complexité de la combinaison des solutions partielles, nous a conduit à développer une variante de la décomposition arborescente : la décomposition linéaire d'un graphe, pour laquelle l'arbre se réduit à un chemin.

### Décomposition Linéaire

Lorsque l'arbre d'une décomposition arborescente se réduit à un chemin, on parle donc de *décomposition linéaire* et de *largeur linéaire* du graphe que l'on note  $pwd(G)$  et qui vérifie  $pwd(G) \leq twd(G)$ . Le facteur exponentiel de la méthode est donc plus faible dans une décomposition arborescente que dans une décomposition linéaire, mais l'implémentation et la gestion de la mémoire présente moins de difficultés. En effet, pour une décomposition arborescente toutes les solutions partielles des  $N_i$  fils doivent être générées et stockées en mémoire en même temps alors que la décomposition linéaire agit comme si une seule feuille était considérée à la fois. La combinaison des solutions partielles est complexe et couteuse pour une décomposition arborescente alors qu'elle s'opère de manière itérative, sommet par sommet dans l'implémentation que nous avons faite de la décomposition linéaire. De plus, tous les principes généraux de la décomposition linéaire sont les mêmes que ceux de la décomposition arborescente. La définition des solutions partielles est la même.

**Principe de la décomposition linéaire pour la fiabilité des réseaux** Les sommets du graphe sont numérotés de manière linéaire par une procédure de voisinage de type BFS. La méthode de résolution est ensuite basée sur une insertion séquentielle des sommets suivant cet ordre, afin de construire les nœuds successifs de la décomposition, et d'en évaluer les solutions partielles. Lorsque le dernier sommet est inséré, le problème est résolu.

**Définition 10 (Numérotation linéaire et ensemble frontière)** *Une numérotation linéaire des sommets de  $G$  est une bijection  $\mathcal{N} : V \rightarrow \{1, \dots, n\}$ . On note  $v(i)$  le sommet numéroté par  $i$ . Alors, l'ensemble  $F_i$  relatif au sommet  $v(i)$  et à la numérotation  $\mathcal{N}$ , défini par l'équation 1.7, est appelé ensemble frontière. La largeur linéaire de cette numérotation, notée  $F_{max}(\mathcal{N})$  est définie par l'équation 1.8. La largeur linéaire d'un graphe correspond à la plus petite largeur linéaire sur l'ensemble des  $n!$  numérotations possibles des sommets. Les schémas de la figure 1.10 montrent l'évolution des ensembles frontières pour l'ordre 1,2,3,4,5,6,7 des sommets. Cet ordre induit un  $F_{max} = 3$ . Or, il existe pour ce graphe qui est un 2-arbre partiel une numérotation de  $F_{max} = 2$  : 3,4,1,2,5,6,7.*

$$F_i = \{v(j) \in V / \exists (v(j), v(l)) \in E \text{ et } j \leq i \leq l\} \tag{1.7}$$

$$F_{max}(\mathcal{N}) = \max\{|F_i|, i = 1, \dots, |V|\} \tag{1.8}$$

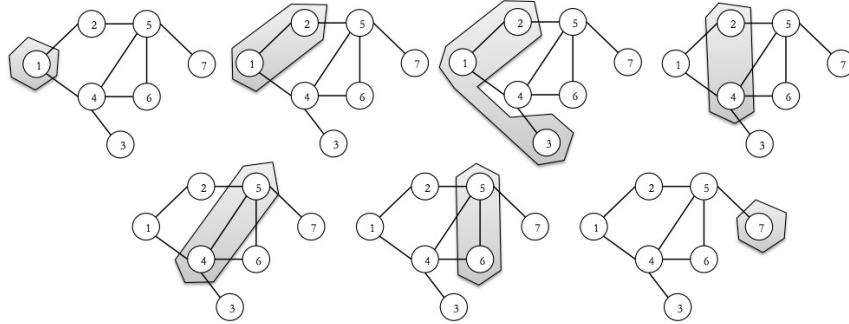


FIGURE 1.10 – Evolution itérative de l'ensemble frontière, de  $F_1$  à  $F_7$

Considérons maintenant le sous-graphe  $G^{-i}$  induit par le sous-ensemble de sommets  $V^{-i} = \{v(1), \dots, v(i)\}$  et son complément  $G^{+i} = G \setminus G^{-i}$ . Alors  $V^{-i}$  et  $V^{+i} \cup F_i$  représentent une décomposition arborescente du graphe, réduite à deux nœuds. La combinaison de leurs solutions partielles permet de résoudre le problème pour  $G$ . La figure 1.11 illustre cette propriété.

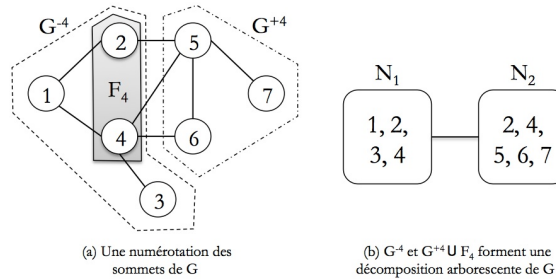


FIGURE 1.11 – Ensemble frontière et décomposition arborescente

Le principe de résolution de la décomposition linéaire que nous avons développée, repose sur cette dernière remarque. On commence par générer et évaluer les solutions partielles de  $G^{-1}$  (un seul sommet  $v(1)$ , une seule solution partielle  $[v(1)]$  pour le problème de fiabilité tous terminaux), puis de  $G^{-2}, G^{-3}, G^{-4}$ , etc., par insertions itératives des sommets  $v(2), v(3), v(4)$ , etc., jusqu'à ce que toutes les solutions partielles de  $G^{-|V|}$  soient générées et évaluées. L'intérêt d'une telle méthode est que toutes les solutions partielles de  $G^{-i}$  peuvent être représentées par les seules partitions de  $F_i$ . Il n'est pas nécessaire de stocker les partitions de  $G^{-i}$ . Les solutions partielles de  $G^{-(i+1)}$  sont générées à partir de celles de  $G^{-(i)}$  et de l'étude des liaisons du sommets  $v(i+1)$  aux sommets de  $F_i$ .

A titre d'exemple, si nous considérons que les sommets du graphe de la figure 1.8 ont été numérotés suivant l'ordre  $a, b, c, \dots, j$  alors la seule solution partielle de  $G^{-1}$  est  $C_1^1 = [a]$  de probabilité 1. Les solutions partielles de  $G^{-2}$  seront obtenues à partir de celle-ci et de  $F_1 = \{a\}$  comme suit :

- $C_1^2 = [ab]$  de probabilité  $1.p_{ab}$
- $C_2^2 = [a][b]$  de probabilité  $1.(1 - p_{ab})$

Les solutions partielles de  $G^{-3}$  seront obtenus à partir de  $C_1^2$ , de  $C_2^2$  et de  $F_2 = \{a, b\}$  comme suit :

- $C_1^3 = [bc]$  obtenue à partir de  $[a b]$  avec au moins une des deux arêtes (a,c) et (b,c) fonctionne, et à partir de  $[a][b]$  avec les deux arêtes (a,c) et (b,c) fonctionnent. La probabilité associée est donc  $p_{ab} \cdot (1 - (1 - p_{ac}) \cdot (1 - p_{bc})) + (1 - p_{ab}) \cdot p_{ac} \cdot p_{bc}$
- $C_2^3 = [b][c]$  obtenue à partir de  $[a b]$  avec (a,c) et (b,c) défailtantes et à partir de  $[a][b]$  avec (a,c) fonctionne et (b,c) défailtante. La probabilité associée est donc  $p_{ab} \cdot (1 - p_{ac}) \cdot (1 - p_{bc}) + (1 - p_{ab}) \cdot p_{ac} \cdot (1 - p_{bc})$

On remarquera que l'ensemble frontière  $F_3$  est réduit à  $\{b, c\}$ . Le sommet a n'ayant pas de voisins avec un numéro d'ordre supérieur à trois, celui-ci n'a plus d'influence sur la génération des solutions partielles des  $G^{-4}, G^{-5}, \dots, G^{-10}$ . La fiabilité tous terminaux du réseau correspond à la probabilité associée la solution partielle  $C_1^n = [j]$  de  $G^{-10}$ .

Nous avons dans nos travaux considéré différentes fonctions de communication, les classiques K-terminaux, 2-terminaux et tous terminaux, mais aussi la 2-arête-connexité et la 2-connexité. Pour chacune de ces fonctions la modélisation de base reste les partitions des sommets de l'ensemble frontière, différemment déclinées en fonction de l'information supplémentaire à représenter. Ainsi pour la fiabilité K-terminaux, il est nécessaire d'indiquer la présence de sommets terminaux dans les composantes connexes. Pour un ensemble frontière  $F_i = \{a, b\}$  les solutions partielles seront  $[ab], [ab]^K, [a][b], [a]^K[b], [a][b]^K, [a]^K[b]^K$ . Plus la fonction est complexe, plus le nombre d'informations nécessaires à sa représentation est grand et par conséquent plus il y a de solutions partielles à considérer.

### 1.3.3 Synthèse des résultats

Le problème d'optimisation consistant à trouver une décomposition sous forme de chemin reste un problème NP-difficile. Néanmoins, l'implémentation est plus simple : la construction heuristique de la décomposition se fait à partir d'une numérotation des sommets du graphe et la gestion des solutions partielles est incrémentale. Nous avons proposé plusieurs heuristiques comme le recuit simulé ou la numérotation dynamique pour produire une décomposition linéaire de largeur réduite, et énoncé quelques propriétés liant la numérotation des sommets à la complexité de l'algorithme [1].

Nous avons proposé un ensemble d'opérateurs permettant la génération et l'évaluation des solutions partielles pour le calcul de la fiabilité des réseaux assurant différentes fonctions de communication. Nous avons testé ces algorithmes sur des instances de la littérature et des graphes aléatoires. Nous avons pu résoudre ces problèmes pour des familles de graphes dont la largeur linéaire n'excédait pas  $k=15$ . Lorsque la largeur linéaire du graphe est petite ( $k=3,4,5$ ), des graphes de très grande taille (plus de 1000 nœuds) peuvent être traités pour des fonctions simples comme la fiabilité tous-terminaux, 2-terminaux et K-terminaux.

## 1.4 Diagrammes de Décision Binaires

Un graphe de décision binaire ou diagramme de décision binaire (ou BDD pour Binary Decision Diagram) est une structure de données efficace pour représenter des fonctions booléennes. Ils furent introduits par Akers en 1978 [34]. En imposant certaines restrictions sur les BDD, Bryant en 1986 [35] a introduit les Diagrammes de Décision Binaires Ordonnés (OBDD) qui ont permis une canonicité dans la représentation des fonctions booléennes. Comme les BDD sont des graphes acycliques dont la construction est basée sur la décomposition de Shannon, les opérations sur les fonctions booléennes peuvent être implémentées comme des algorithmes de graphe opérant sur les BDD. Cette représentation, très compacte pour la grande majorité des fonctions booléennes, permet de les manipuler efficacement. L'avantage d'une telle représentation est qu'elle permet le traitement en une seule fois de sous-problèmes équivalents grâce à sa canonicité et son partage

de sous-graphes. Les BDD trouvent leurs applications dans différents domaines [36], comme la vérification des circuits logiques, la vérification des machines séquentielles, l'optimisation logique des circuits combinatoires, la simulation symbolique et la fiabilité des réseaux.

Le fonctionnement d'un réseau peut être représenté par une fonction booléenne dont les variables représentent l'état de fonctionnement et de défaillance des différents composants du réseau. Cette fonction n'est autre qu'une expression des chemins minimaux et/ou des coupes minimales. L'utilisation des BDD dans le domaine de la fiabilité a été introduite par Madre et Coudert [37] et par A. Rauzy [38]. Ces travaux ont permis, pour la première fois, de traiter qualitativement et quantitativement de grands arbres de défaillance en quelques secondes. Les arbres de défaillance, outils majeurs dans les études de fiabilité, représentent graphiquement les combinaisons d'événements conduisant à la réalisation de l'événement indésirable [39]. Les BDD sont, à ce jour, largement utilisés pour différentes applications, dont la fiabilité des réseaux. Ce qui vaut à nos travaux sur les BDD, d'être largement cités dans la littérature récente notamment l'article "K-Terminal Network Reliability Measures With Binary Decision Diagrams" [3] dont Researchgate comptabilise 122 citations dont 3 en 2019.

### 1.4.1 Définitions des BDD généralités

Le diagramme de décision binaire permet de représenter de manière très compacte des fonctions booléennes. Sa construction est basée sur la réduction de l'arbre de décision (arbre d'états) après un étiquetage des nœuds de celui-ci. Cette réduction consiste à fusionner les sous-arbres équivalents et à éliminer des sous-arbres inutiles.

**Définition 11 (BDD)** *Un BDD est un graphe acyclique orienté (aussi appelé DAG : Direct Acyclic Graph) basé sur la décomposition de Shannon. Il vérifie les propriétés suivantes :*

- *Le graphe a deux feuilles étiquetées par 0 et 1.*
- *Chaque nœud interne  $v$  est étiqueté par une variable booléenne  $x_i$  (noté  $\text{var}(v)$ ) et possède deux arêtes sortantes. Chacune de ces arêtes représente une valeur de  $x_i$ .*

*On note par  $\text{low}(v)$  et  $\text{high}(v)$  les deux fils de  $v$  pointés respectivement par les arêtes sortantes étiquetées par 0 et 1.*

**Définition 12 (Chemin-calcul)** *On appelle chemin-calcul d'une entrée  $a = (a_1, \dots, a_n) \in \mathbb{B}^n$  le chemin partant de la racine du BDD, et empruntant les arêtes correspondant aux valeurs des  $a_i$  :  $(v, \text{high}(v))$  si  $a_i = \text{var}(v) = 1$  ;  $(v, \text{low}(v))$  si  $a_i = \text{var}(v) = 0$ .*

**Définition 13** *Un BDD représente une fonction booléenne  $f : \mathbb{B}^n \rightarrow \mathbb{B}$  si pour chaque entrée  $a = (a_1, \dots, a_n) \in \mathbb{B}^n$ , le chemin-calcul associé mène à une feuille étiquetée par  $f(a)$ .*

**Définition 14 (Largeur d'un BDD)** *La taille d'un BDD est le nombre de ses nœuds internes. Si  $n_i$  représente le nombre de nœuds sur le niveau  $i$  (les nœuds étiquetés par la variable booléenne  $x_i$ ) alors la largeur  $W$  du BDD est définie par :  $W = \max\{n_i, i = 1, \dots, n\}$ .*

**Définition 15 (OBDD)** *Soit  $<$  un ordre total sur les variables booléennes  $x_1, x_2, \dots, x_n$ . Un BDD est ordonné (OBDD) si pour tout couple de nœuds  $u$  et  $v$  du BDD, respectivement étiquetés  $x_i$  et  $x_j$ , alors s'il existe un chemin allant de  $u$  à  $v$  cela implique que  $x_i < x_j$ .*

Ainsi, tous les nœuds qui sont à la même distance de la racine sont étiquetés par la même variable booléenne. L'exemple de la figure 1.12 donne un exemple de BDD ordonné suivant l'ordre des variables  $x_1 < x_2 < x_3 < x_4$ . Un OBDD optimal ou minimal est le OBDD dont la taille est la plus petite parmi les  $n!$  OBDD possibles ; l'ordonnement correspondant est dit optimal. Trouver un ordre optimal des variables est un problème NP-Difficile.

**Définition 16 (ORBDD)** *Un OBDD est réduit (ORBDD) s'il vérifie les deux conditions suivantes :*

- *Chaque nœud représente une expression booléenne différente (i.e. il n'existe aucun sous-arbres isomorphes dans le OBDD).*
- *Chaque nœud a des fils différents (pour un nœud  $u$  quelconque, on a  $\text{low}(u) \neq \text{high}(u)$ ).*

Ainsi sur la figure 1.12, le ORBDD représente la fonction booléenne  $f = (x_1 \iff x_3) \wedge (x_2 \iff x_4)$  pour l'ordre des variables suivant :  $x_1 < x_2 < x_3 < x_4$ . Les flèches en pointillés correspondent à l'instanciation de la variable à 1 et les flèches pleines à l'instanciation de la variable à 0. Sur cette figure, (1, 3)(3, 5)(5, 11) est un chemin-calcul du ORBDD, allant de la racine 1 à la feuille 11. Les valeurs prises par les variables booléennes sur ce chemin sont :  $x_1 = 0$ ;  $x_2 = 1$ ;  $x_3 = 0$  et  $x_4$  indifférenciée.  $f(0, 1, 0, -) = 0$  car la feuille 0 est atteinte.

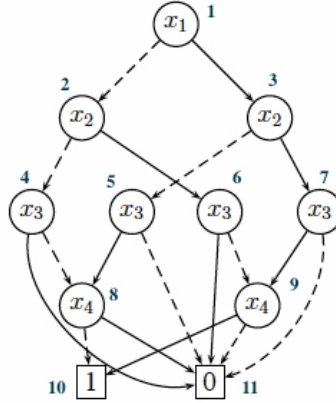


FIGURE 1.12 – BDD de la fonction booléenne  $f = (x_1 \iff x_3) \wedge (x_2 \iff x_4)$

La difficulté de manipulation des ORBDD est évidemment l'identification des expressions booléennes en chacun des nœuds, afin de fusionner les redondances et éliminer les nœuds stériles. Les nombreuses bibliothèques décrites par la littérature, proposent des solutions de stockage des ORBDD, basée sur l'utilisation de tables de hachages (<https://www.cs.cmu.edu/modelcheck/bdd.html>).

Dans la thèse de Gary Hardy nous avons décrit l'ensemble des opérations utilisées pour transformer un BDD en un ORBDD. De nombreuses recherches ont été effectuées afin de transformer les concepts de BDD en une implémentation machine efficace en temps et en mémoire [35]. L'idée principale repose sur un étiquetage des nœuds et pour chacun d'entre eux un référencement sur les deux fils (*low et high*) à travers l'utilisation des techniques de tables de hachages.

**Remarque :** Dans ce document comme dans la littérature, le terme BDD désignera en réalité un ORBDD.

### 1.4.2 Modélisation de la fiabilité des réseaux par les BDD

Dans le domaine de la fiabilité de réseaux, Trivedi [40] et Sekine et al. [41] ont montré pour la première fois comment construire efficacement le BDD encodant la fiabilité tous-terminaux et 2-terminaux. En 2002, Yeh et al. [42] proposent une méthode plus générale, pour représenter la fiabilité K-terminaux par un BDD. Les approches par BDD permettent de minimiser l'impact de l'explosion du nombre d'états du système grâce à une représentation concise de ces états. Leur utilisation en fiabilité de réseau reste aujourd'hui l'objet de nombreuses études [43, 44]. L'intérêt de l'emploi des BDD pour la fiabilité est le stockage de l'information, de la structure du réseau, sur laquelle il est possible de revenir pour obtenir des caractéristiques intéressantes comme le facteur d'importance des composants sur la fiabilité globale du réseau.

Lors de la thèse de Gary Hardy, nous avons développé un premier algorithme permettant de résoudre le problème de fiabilité K-terminaux, pour  $2 \leq K \leq n$ . Cet algorithme, nommé BDDPart, permet d'encoder le BDD correspondant et de calculer ensuite la fiabilité par l'évaluation des probabilités associées aux chemins-calculs de la racine à la feuille 1.

L'algorithme BDDPart est une généralisation de l'algorithme pour le calcul spécifique de la fiabilité tous-terminaux que nous présentons dans l'article [45]. Nous nous limitons, par soucis de

clarté, d'abord à la fiabilité de réseaux avec sommets parfaits mais les différents cas ont été traités dans nos travaux (les sommets comme les liens peuvent être sujets à des défaillances).

**Définition 17 (Fonction de structure)** *La fonction de structure d'un réseau peut être représentée par la fonction booléenne  $\phi \rightarrow \{0, 1\}$  définie comme suit :*

- $\phi(x_1, x_2, \dots, x_m) = 1$  si le réseau fonctionne
- $\phi(x_1, x_2, \dots, x_m) = 0$  sinon

Le but est d'encoder cette fonction par une structure de BDD. Une fois la fonction de fiabilité encodée, la valeur de la fiabilité du réseau est calculée à partir du BDD suivant la formule récursive suivante :

- $R_K(G) = Pr\{\phi = 1\}$
- $R_K(G) = p_i \cdot Pr\{\phi_{x_i=1} = 1\} + q_i \cdot Pr\{\phi_{x_i=0} = 1\}, \forall i \in \{1, \dots, m\}$  :

**Principe de construction du BDD :** Le principe générale de la méthode de construction du BDD à partir du graphe stochastique se base sur la contraction et la suppression d'arêtes du graphe, suivant leur état de fonctionnement et de défaillance : l'arête est contractée si elle fonctionne, ce qui correspond à une liaison *high* dans le BDD, et elle est supprimée si elle est défaillante, ce qui correspond à une liaison *low* dans le BDD. Si  $e_i$  est une arête du graphe  $G$ , on note  $G*e_i$  le graphe où  $e_i$  est contractée et  $G \setminus e_i$  le graphe où  $e_i$  est supprimée. Alors la fiabilité est calculée par la formule 1.9 suivante :

$$f(x) = \begin{cases} p_i \cdot R_K(G*e_i) & \text{si } e_i \text{ est un isthme actif} \\ R_K(G \setminus e_i) & \text{si } e_i \text{ est une boucle} \\ (1 - p_i) \cdot R_K(G \setminus e_i) + p_i \cdot R_K(G*e_i) & \text{sinon} \end{cases} \quad (1.9)$$

Un isthme actif est une arête dont la suppression augmente le nombre de composantes connexes et dont les  $K$  sommets se trouvent dans des composantes différentes. Une boucle est une arête ayant des extrémités identiques. Le calcul de la formule précédente peut être représentée par un arbre binaire tel que : la racine correspond au graphe original  $G$  et chaque nœud représente les graphes mineurs de  $G$ . On parle d'*arbre d'expansion*, généré à partir d'un ordre fixé des arêtes. La figure 1.14 montre le graphe d'expansion obtenu pour l'ordre  $e_1, e_2, e_3, e_4, e_5, e_6$  des arêtes dans le cas de la fiabilité  $K$ -terminaux pour le graphe de la figure 1.13.

C'est à partir de cet arbre d'expansion que sera codé le BDD. Les nœuds du  $i$ -ème niveau représentent tous les graphes mineurs que l'on peut obtenir à partir de  $G$  en supprimant/contractant toutes les arêtes  $e_k$  avec  $k < i$ .

Le problème posé par ces arbres d'expansion est le grand nombre potentiel de graphes isomorphes représentés en chacun des nœuds d'un même niveau. La solution apportée à ce problème, repose comme nous l'avions introduit pour le problème de décomposition arborescente, sur une représentation sous forme de partitions de l'état de connexité des sommets dont les arêtes incidentes ont été soit supprimées, soit contractées. Comme pour les solution partielles de la décomposition linéaire, ces états de connexion se réduisent aux états de connexion des *sommets de l'ensemble frontière du graphe*. Celui-ci est défini lors du processus de construction de l'arbre d'expansion.

**Définition 18 (Ensemble frontière et BDD)** *Pour chaque niveau  $i$  de l'arbre d'expansion, on définit l'ensemble  $E_i$  des arêtes  $e_j$  pour lesquelles l'état est fixé ( $j < i$ ) et l'ensemble  $\bar{E}_i$  des arêtes  $e_k$  pour lesquelles l'état n'est pas encore fixé ( $k \geq i$ ). Alors l'ensemble frontière est défini comme suit :*

$$F_i = \{v \in V \mid \exists (u, v) \in E_i \text{ et } \exists (v, w) \in \bar{E}_i\}$$

Tous les graphes mineurs de l'arbre d'expansion peuvent donc être modélisés par une partition de l'ensemble frontière  $F_i$  pour  $i = 1, \dots, m$ . Ces partitions modélisent l'état de connexion des sommets de  $F_i$  à travers le sous-graphe induit par l'état des arêtes de  $E_i$ . La figure 1.14 correspond à l'arbre d'expansion de la figure 1.13 et la figure 1.15 au BDD associé. L'algorithme BDDPart que nous avons proposé, repose sur le théorème 1, que nous avons démontré dans [3].



**Théorème 1 (Graphes isomorphes)** Soient  $G_1$  et  $G_2$  deux sous-graphes de  $G$  obtenus par suppressions et/ou contractions successives des arêtes de  $E_i$ . Les sous-graphes  $G_1$  et  $G_2$  sont isomorphes si et seulement si leurs partitions respectives associées à  $F_i$  sont identiques.

Les sous-graphes de l'arbre d'expansion sont donc représentés par les partitions des  $m$  ensembles  $F_i$ . La construction du BDD se fait niveau par niveau en partant de la racine. Le niveau  $k + 1$  est construit à partir du niveau  $k$  ( $1 \leq k < m - 1$ ). La construction du BDD se résume donc à construire les partitions de  $F_i$ , à partir des partitions de  $F_{i-1}$  et de la contraction et/ou la suppression de l'arête  $e_i$ . Chaque nœud du BDD est donc étiqueté par un numéro d'ordre de partition comme utilisé dans la décomposition arborescente. Une table de hachage est utilisée pour stocker les numéros des partitions de chaque sous-graphe. Si l'arbre d'expansion génère deux partitions identiques pour un même ensemble frontière, celles-ci seront étiquetées par le même entier et donc fusionnées fusionnées pour ne former qu'un seul nœud du BDD. La figure 1.15 représente la génération des différentes partitions et le BDD associé au graphe de la figure 1.13, le numéro d'ordre de chaque partition apparaît entre parenthèses.

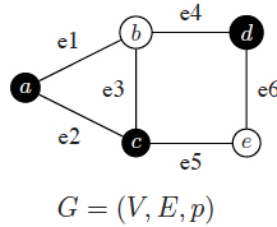


FIGURE 1.13 – Graphe : les sommets noircis sont les sommets terminaux.

L'ordre de traitement des arêtes lors de la génération du BDD a des conséquences directes sur la largeur du BDD, c'est-à-dire sur le nombre maximum de nœuds sur un même niveau. La largeur du BDD de la figure 1.15 vaut deux. Or, la complexité de notre algorithme est exponentielle en fonction de cette largeur. En effet, le nombre maximum de partitions à considérer par niveau est exponentiel en fonction du nombre de sommets des ensembles frontières. Nous avons montré lors des travaux de thèse de Gary Hardy, que la largeur minimale du BDD est égale à la largeur linéaire  $F_{max}$  du graphe stochastique représentant le réseau. La complexité de BDDPart est donc en  $O(m \times \exp(F_{max}))$ .

**Mesure d'importance et optimisation de la conception de réseaux** Grâce à la structure des BDD, il est possible de calculer le facteur d'importance de manière très efficace, pour chaque composant du système. Ce facteur d'importance donne la mesure de l'impact du fonctionnement de chaque composant sur la fiabilité du réseau. Cette notion fut introduite par Birnbaum [46] (définition 19), et améliorée pour devenir le facteur d'importance critique (définition 20).

**Définition 19** Soit  $e_k$  une arête du réseau modélisé par  $G = (V, E, p)$ . Alors le facteur d'importance de  $e_k$  est défini comme suit :

$$\begin{aligned} \mathcal{I}_k^B &= R(G/e_k \text{ fonctionne}) - R(G/e_k \text{ défaille}) \\ \mathcal{I}_k^B &= R(G_{*e_k}) - R(G_{\setminus e_k}) \end{aligned}$$

**Définition 20** Soit  $e_k$  une arête du réseau modélisé par  $G = (V, E, p)$ . Alors le facteur d'importance critique de  $e_k$  est défini comme suit :

$$\mathcal{I}_k^C = \mathcal{I}_k^B \frac{p_k}{R(G)}$$

Grâce à ces mesures, les composants peuvent être discriminés : plus le facteur d'importance est élevé, plus la fiabilité du composant participe à la fiabilité du réseau. Nous avons proposé un algorithme permettant le calcul des facteurs d'importance critiques en ne parcourant qu'une seule

fois le BDD. Ce qui nous a permis de proposer une heuristique efficace *NDImprovement* pour résoudre un problème d'optimisation de conception de réseaux.

Il s'agit d'un problème d'optimisation de topologie de réseau (chaque composant possède un coût d'utilisation) avec contrainte de fiabilité : quels sont les nœuds du réseau et comment les interconnecter afin que la fiabilité globale atteigne une fiabilité minimale de  $R_0$  à moindre coût ? L'heuristique que nous avons proposée, repose sur un principe d'élimination des composants dont le facteur d'importance est le plus faible possible et de réparation afin de maintenir une fiabilité supérieure à  $R_0$  [47].

### 1.4.3 Synthèse des Résultats

Nous avons proposé de nouveaux algorithmes *BDDPart*, permettant de calculer la fiabilité K-terminaux pour  $1 < |K| < |V|$ , s'appuyant sur la représentation de la fonction de structure de réseau par les BDD. Nous avons également traité le cas où les nœuds du réseau sont susceptibles de défaillance.

Contrairement aux autres méthodes également basées sur les BDD, notre algorithme a une complexité en temps indépendante de la taille de l'ensemble  $K$  des sommets terminaux et indépendante de la taille du réseau. Elle dépend uniquement de la largeur-linéaire du réseau grâce à la représentation des sous-grphes isomorphes sous forme de partitions. Les résultats expérimentaux obtenus sur différents benchmarks de la littérature montrent l'efficacité de cette approche.

Nous avons testé trois catégories de graphes :

- Les graphes complets pour lesquels  $F_{max} = |V - 1|$ . Pour ces graphes nous avons pu encoder les BDD correspondants jusqu'à  $F_{max} = 15$
- Les grilles  $L \times C$  sont des graphes réguliers pour lesquels  $F_{max} = \min\{L, C\}$  et  $|V| = L \times C$ . Les BDD de grilles de plusieurs milliers de nœuds ont pu être encodés pour des largeurs modérés ( $F_{max} \leq 21$ )
- Les graphes de la littérature étaient peu nombreux. Nous avons comparé notre approche à d'autres méthodes. Comme ces graphes sont de largeur linéaire petite ( $F_{max} \leq 6$ ) nos algorithmes se sont montrés beaucoup plus efficaces en temps avec en moyenne un facteur 10.

En ce qui concerne le problème d'optimisation de conception de réseau, nous nous sommes comparés à l'algorithme génétique *LS/NGA* de Altıparmak & al. [48]. Nos résultats améliorent drastiquement les résultats de la littérature de l'époque. Sur 42 instances *NDImprovement* fournit des solutions qui sont en moyenne à 3% de l'optimalité et à 0% pour 14 instances. Quant au temps de calcul ils présentent un rapport de 100 en notre faveur. Ces très bons résultats montrent à quel point les BDD sont des outils très puissants pour ce type de problème.

## 1.5 Conclusion

Le problème de Fiabilité des Réseaux m'a été proposé par le Professeur Jacques Carlier comme sujet de thèse de 1990 à 1993, afin de faire suite aux travaux de thèse d'Olympia Theologou au laboratoire HeuDiaSyC de l'UTC. Il s'agissait alors de développer des algorithmes permettant d'utiliser une décomposition linéaire du graphe pour le calcul de fiabilités des réseaux et d'en tester les caractéristiques et les limites. Nous avons poursuivi nos travaux à travers le co-encadrement de la thèse de Jean-François Manouvrier de 1995 à 1998 sur une généralisation des fonctions de communication pour le problème de fiabilité des réseaux ainsi que d'une manière plus générale sur la résolution de différents problèmes combinatoires comme le problème du voyageur de commerce et le problème des arbres minimaux de Steiner, par une décomposition linéaire. Tous les résultats issus de ces travaux ont clairement montrés que certaines familles de graphes, dont la largeur linéaire n'est pas trop grande, ont des caractéristiques permettant de développer des algorithmes polynomiaux pour résoudre certains problèmes combinatoires, comme celui de la fiabilité des réseaux.

Le professeur Nikolaos Limnios du Laboratoire de Mathématiques Appliquées de Compiègne, ayant connaissance de nos travaux, put remarquer une certaine correspondance entre les méthodes de décomposition que nous développons et les diagrammes de décision binaires (BDD) utilisés par les fiabilistes pour évaluer les systèmes complexes. Nous avons donc collaboré lors du co-encadrement de la thèse de Gary Hardy de 2004 à 2007 sur l'utilisation des BDD pour le calcul de la fiabilité des réseaux. Nous avons pu par cette étude, combiner la puissance de représentation des BDD, à la concision de représentation de l'information permise par la modélisation utilisée dans la décomposition linéaire. Nous avons pu montrer que la largeur des BDD étaient inférieurement bornée par la largeur linéaire du graphe. Nous avons également développé une heuristique permettant de résoudre un problème d'optimisation de conception de réseau sous contrainte de fiabilité. Pour ce problème aussi, les BDD ont montré d'excellents résultats.

L'ensemble de ces travaux ont été publiés dans des revues internationales et des conférences internationales de haut rang. Les trois articles ci-dessous mentionnés se trouvent dans les annexes de ce mémoire.

- [1] Jacques Carlier and Corinne Lucet. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics*, 65(1) :141-156, 1996.
- [2] Corinne Lucet, Jean-Francois Manouvrier, and Jacques Carlier. Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs. *Algorithmica*, 27(3) :316-336, 2000.
- [3] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Trans. Reliability*, 56(3) :506-515, 2007

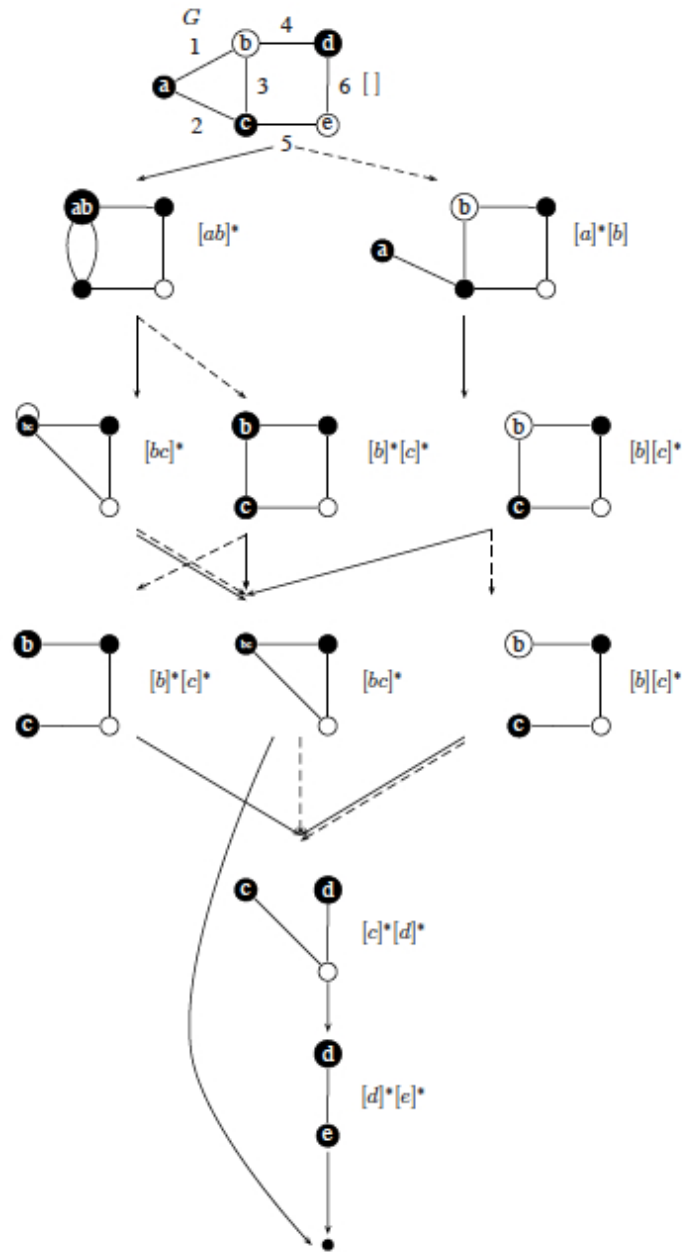


FIGURE 1.14 – Arbre d’expansion du graphe de la figure 1.13 pour l’ordre des arêtes suivant :  $e_1, e_2, e_3, e_4, e_5, e_6$ .

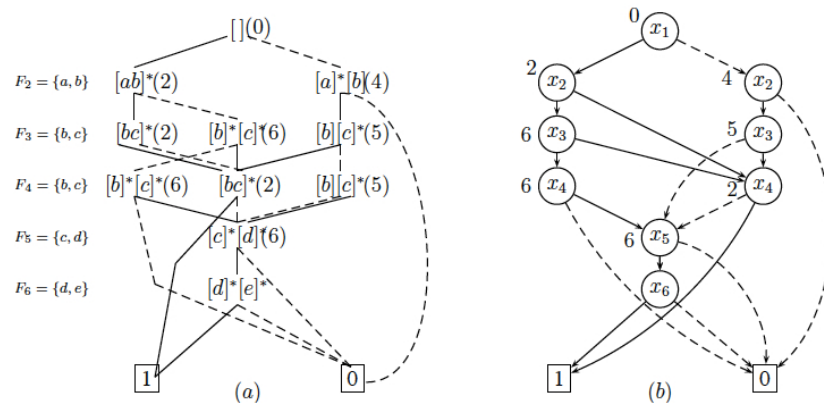


FIGURE 1.15 – Arborescence des partitions et BDD associé

# Chapitre 2

## Problèmes de coloration de graphes

### Sommaire

---

<b>2.1</b>	<b>Problème du nombre chromatique d'un graphe . . . . .</b>	<b>34</b>
2.1.1	Introduction . . . . .	34
2.1.2	Définition du problème . . . . .	34
2.1.3	Décomposition arborescente pour le problème du nombre chromatique . . . . .	35
2.1.4	Décomposition linéaire pour le problème du nombre chromatique . . . . .	36
2.1.5	Pré-traitements . . . . .	38
2.1.6	Synthèse des résultats sur la coloration . . . . .	38
<b>2.2</b>	<b>Somme chromatique et force d'un graphe . . . . .</b>	<b>39</b>
2.2.1	Introduction . . . . .	39
2.2.2	Définition du problème . . . . .	40
2.2.3	Méthodes approchées . . . . .	41
2.2.4	Formulation SAT . . . . .	47
2.2.5	Synthèse des résultats WP-MaxSAT et WP-MinSAT . . . . .	50
2.2.6	Représentation de l'espace des solutions par Motifs . . . . .	50
2.2.7	Synthèse des résultats Motifs et bornes supérieures de la force . . . . .	54
2.2.8	Synthèse des résultats Motifs et borne inférieure de la somme chromatique : . . . . .	55
<b>2.3</b>	<b>Conclusion . . . . .</b>	<b>56</b>

---

Les problèmes de coloration de graphes ont été intensivement étudiés dans la littérature et ce depuis de nombreuses décennies. Historiquement ce problème est né d'un problème pratique de coloration de la carte des comtés d'Angleterre par le mathématicien et botaniste Francis Guthrie en 1852. Celui-ci est le premier à poser le célèbre problème des quatre couleurs. Depuis de nombreux problèmes réels, comme des problèmes d'affectation de ressources, ou des problèmes d'ordonnement de tâches, ont pu être modélisés comme des problèmes de coloration de graphes. Les couleurs dans ces différents cas, représentent des coûts ou des créneaux de temps associés aux entités d'un système représenté par un graphe.

Les problèmes de coloration, dans le cas général, sont des problèmes NP-difficiles, pour lesquels les méthodes exactes qui calculent une solution optimale, ne peuvent traiter que des instances de tailles modestes. En conséquence, la littérature est riche de nombreuses méthodes approchées dédiées à la résolution de ces problèmes pour de grandes instances. Les problèmes de coloration de graphes se différencient les uns des autres par le type de graphe, orienté ou non orienté, par les entités du graphe sur lesquelles sont associées les couleurs, les sommets et/ou les arêtes, par les contraintes relatives aux couleurs et enfin par la nature de la fonction objectif. Mes travaux se sont concentrés sur deux problèmes de coloration de graphe : le problème du nombre chromatique d'un graphe, et le problème de la somme chromatique d'un graphe. Pour chacun de ces problèmes nous avons proposés des méthodes exactes et approchées et nous avons utilisé les benchmarks de

références COLOR02 et DIMACS, afin de nous comparer à l'ensemble des méthodes existantes.

Dans ce chapitre est décomposé en deux parties, la première traite du problème du nombre chromatique d'un graphe et la seconde de la somme chromatique d'un graphe. Pour chacun d'entre eux nous donnons les définitions du problème, le contexte de l'étude et nos travaux associés.

Pour le problème du nombre chromatique nous faisons le lien entre ce problème et la fiabilité des réseaux à travers la définition des solutions partielles pour une résolution par la décomposition linéaire du graphe. Nous exposons ensuite les différentes améliorations que nous avons proposées, comme les opérateurs de réduction du graphe et l'introduction de bornes dans le processus de décomposition.

Pour le problème de la somme chromatique, nous présentons succinctement les deux approches utilisées pour sa résolution. Premièrement une approche heuristique avec la proposition des algorithmes gloutons MDSAT et MRLF, d'un algorithme de recherche locale IDCA et d'un algorithme mémétique MA-MSCP. Ensuite, la modélisation de ce problème sous forme de problème Weighted Partial MaxSAT et Weighted Partial MinSAT sont montrées et comparées pour une approche de résolution exacte, utilisant les solveurs de la littérature. Enfin pour terminer ce chapitre, nos derniers travaux sur la représentation de l'espace des solutions par "motifs" sont décrits.

Pour chaque partie, une synthèse des résultats est présente. Tous les détails ne sont pas donnés ici, mais les principaux articles référents se trouvent dans la partie annexe du document.

## 2.1 Problème du nombre chromatique d'un graphe

### 2.1.1 Introduction

Après nos travaux sur la décomposition linéaire pour la résolution du problème de la fiabilité des réseaux, nous souhaitons étudier cette technique pour d'autres problèmes de graphe. Nous avons donc choisi le problème du nombre chromatique et cherché le moyen de financer cette étude en répondant à un appel à projet du Conseil Régional de Picardie. Ainsi, les travaux entrepris autour du problème du nombre chromatique se sont déroulés dans le cadre du projet de recherche 2000.3 du Pôle Modélisation, en collaboration avec le Professeur Jacques Carlier et le Professeur Aziz Moukrim du Laboratoire HeuDiaSyC de Compiègne (UTC). Celui-ci a permis le financement de la thèse de Mme Florence Mendes (2005), actuellement Maître de Conférence à l'Université de Bourgogne. Notre but était alors d'utiliser la décomposition linéaire, combinée à des techniques de type branch-and-bound afin de réduire le nombre de solutions partielles à énumérer pour traiter les graphes tests de la littérature et nous mesurer aux meilleures méthodes publiées. Ces travaux ont été publiés dans le journal international *Computers & Operations Research* [49] et dans des conférences internationales [50] et nationales (ROADEF 2002,2003,2005 et MAJESTIC 2004).

### 2.1.2 Définition du problème

Un graphe non-orienté est un couple  $G = (V, E)$  composé d'un ensemble de sommets  $V$  et d'un ensemble d'arêtes  $E \subseteq V \times V$ . On notera  $n = |V|$  et  $m = |E|$ . Une coloration du graphe  $G$  est l'affectation d'une couleur  $c(x)$  à chaque sommet  $x$  telle que  $c(x) \neq c(y) \forall (x, y) \in E$ . Elle peut également être représentée comme une partition de  $V$  en  $k$  parties,  $X = \{X_1, X_2, \dots, X_k\}$ , de telle sorte qu'il n'existe pas d'arête entre deux sommets d'une même partie, i.e.  $\forall i = 1, 2, \dots, k, \forall (u, v) \in X_i \times X_i$  alors  $(u, v) \notin E$ . Tous les sommets de  $X_i$  partagent la même couleur.  $X_i$  est également appelée *classe couleur*. Si le nombre de couleurs utilisées est  $|X| = k$ , la coloration de  $G$  est appelée une  $k$ -coloration. Le problème de  $k$ -coloration consiste, à répondre à la question : «  $G$  est-il coloriable avec seulement  $k$  couleurs ? ». Le problème de coloration de graphe est le problème d'optimisation associé au problème de  $k$ -coloration. Il consiste à trouver la valeur minimale de  $k$  pour laquelle une  $k$ -coloration est possible. Cette valeur est appelée nombre chromatique du graphe et est notée  $\chi(G)$ .

De nombreux problèmes difficiles comme l'allocation de fréquences en téléphonie ou les problèmes d'emploi du temps et d'ordonnancement de tâches, peuvent se modéliser par un problème de coloration de graphe. Pour le problème d'allocation de fréquences, les sommets représentent les entités

du système comme une *station de base* et les arêtes correspondent aux relations entre ces entités (incompatibles, voisins, etc.). Afin de définir des classes d'entités indépendantes, on associe à chaque sommet une « couleur ». Deux sommets partageant la même couleur pourront par exemple partager la même fréquence, sans crainte d'interférences.

Les recherches menées sur le problème du nombre chromatique d'un graphe sont de deux types. La première est une approche théorie des graphes, dans laquelle des propriétés structurelles des graphes sont énoncées permettant de faciliter ou de directement déterminer la recherche du nombre chromatique. A titre d'exemple, nous citerons les graphes triangulés pour lesquels la résolution du problème est linéaire. La seconde approche consiste à ne considérer que le cas général, sans propriétés particulières sur les graphes et à développer des algorithmes exactes ou approchés pour déterminer le nombre chromatique dans le premier cas, et une approximation de celui-ci dans le second.

Pour les algorithmes exactes, les principales techniques sont de type branch-and-bound [51, 52, 53], branch-and-cut [54], basées sur la détermination de sous-graphes critiques pour réduire la taille de l'instance [55], ou encore basées sur une formulation en logique propositionnelle SAT [56, 57]. Nos recherches sur le problème du nombre chromatique s'inscrivent dans ce cadre. Mais dans le cas général, ces méthodes exactes ne permettent de résoudre que des instances de petite ou de moyenne taille.

Pour ces raisons, la littérature depuis les années 1970, regorge de travaux présentant des méthodes approchées en concurrence pour traiter les plus grandes des instances de benchmarks de références comme DIMACS ou COLOR02. Succinctement, ces méthodes sont de type glouton, comme les célèbres algorithmes DSATUR [51] et RLF [58], et de type métaheuristique. On trouve des algorithmes tabou [59, 60, 61], des algorithmes de recuit simulé [62], et des algorithmes évolutionnaires [63, 64, 65]. Ces derniers sont toujours très largement utilisés pour la résolution de problèmes de coloration de graphe, et souvent hybridés avec éventuellement des méthodes d'apprentissage [66, 67, 68].

Les problèmes de colorations font toujours l'objet de l'attention de nombreux chercheurs à travers le monde. Une simple recherche sur internet à partir des mots clés « *graph colouring problem* » pour des articles de 2019, liste environ 5500 items.

### 2.1.3 Décomposition arborescente pour le problème du nombre chromatique

Pour résoudre le problème de coloration par décomposition arborescente, le graphe est décomposé, comme présenté dans la section 1.3.1 du chapitre 1, en sous-ensembles de sommets  $N_1, N_2, \dots, N_l$  formant un arbre  $T$  (cf Figure 2.1) et répondant à la définition 5 du chapitre 1. Quelque soit le problème à résoudre, cette partie qui consiste à trouver une décomposition du graphe de largeur la plus petite possible, reste la même. Ce qui varie d'un problème à l'autre est la modélisation des solutions partielles et leurs évaluations.

#### Solutions partielles

Comme pour le problème de la fiabilité des réseaux, les solutions partielles de chaque nœud  $N_i$  de la décomposition arborescente, sont représentées par les partitions des sommets de  $N_i$ . Si pour le problème de fiabilité un bloc d'une partition représentait une composante connexe du graphe, pour le problème de coloration un tel bloc représente un ensemble de sommets de même couleur. Dans le cas de la coloration de graphe, une partition est valide si elle représente une coloration valide des sommets de  $N_i$ , c'est-à-dire une partition pour laquelle les sommets d'un même bloc forment un ensemble stable du graphe. La valeur d'une solution partielle d'un nœud  $N_i$  représentée par la partition  $P = B_1, B_2, \dots, B_r$ , est le nombre minimum de couleurs nécessaires pour colorier le sous-graphe  $G(T(N_i))$ , induit par l'ensemble des sommets représentés dans le sous-arbre  $T(N_i)$  de racine  $N_i$  avec la contrainte que les sommets de  $N_i$  soient coloriés selon cette partition  $P$ . Sur l'exemple de la figure 2.1, la valeur de la partition  $[bc][d]$  représente le nombre de couleurs minimum pour colorier le sous-graphe induit par les sommets de  $N_2 \cup N_4 \cup N_5$  avec la contrainte : « b et c



sont de même couleur et d est d'une autre couleur ».

### Evaluation des solutions partielles

Soient  $P = B_1, B_2, \dots, B_r$  une partition d'un ensemble  $N_i$  et  $N_j$  un ensemble tel que  $N_i \neq N_j$ . On définit l'intersection entre  $P$  et  $N_j$  comme étant égale à la partition  $P \cap N_j = B_1 \cap N_j, B_2 \cap N_j, \dots, B_r \cap N_j$ . Alors, les solutions partielles sont évaluées de la façon suivante :

1. Si la partition  $P$  n'est pas valide,  $val(N_i, P) = \infty$
2. Si  $N_i$  est une feuille et si la partition  $P$  est valide,  $val(N_i, P) = r$
3. Si  $N_i$  est un noeud interne dont les fils sont  $N_j$  et  $N_k$  et si  $P$  est valide,  $val(N_i, P) = \min\{max(val(N_j, P1), val(Nk, P2), r) \mid P \cap N_j = P1 \cap N_i \text{ et } P \cap N_k = P2 \cap N_i\}$

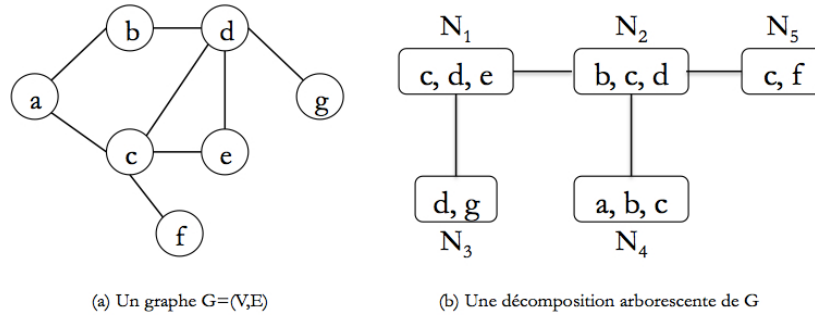


FIGURE 2.1 – Décomposition arborescente

Le tableau de la figure 2.2 illustre une résolution du problème du nombre chromatique par la décomposition arborescente de la figure 2.1. Les noeuds sont évalués dans l'ordre  $N_5, N_4, N_2, N_3$  et enfin la racine  $N_1$ . Le nombre chromatique est donné par le minimum des valeurs associées aux solutions partielles de  $N_1$  et vaut dans ce cas 3.

### 2.1.4 Décomposition linéaire pour le problème du nombre chromatique

Comme exposé dans le chapitre 1, la largeur-arbre d'un graphe est inférieure ou égale à sa largeur-linéaire, et en conséquence le facteur exponentiel de la méthode est plus faible dans une décomposition-arbre que dans une décomposition-linéaire. Pour les mêmes raisons que celles évoquées pour la résolution du problème de fiabilité des réseaux, lors de la thèse de Florence Mendes, nous avons choisi de développer un algorithme de décomposition-linéaire pour résoudre le problème du nombre chromatique.

A partir d'une numérotation linéaire des sommets, les ensembles frontières répondant à la définition 10 du chapitre 1 sont générés de manière itérative suivant l'ordre des sommets. Pour chacun d'entre eux, les solutions partielles, donc les partitions, sont construites et évaluées.

Le principe de résolution de la décomposition-linéaire que nous avons développée lors de la thèse de Florence Mendes, reprend exactement le principe de résolution énoncé pour le problème de fiabilité des réseaux. Evidemment les solutions partielles même si celles-ci partagent une même structure, ne sont pas identiquement évaluées.

A titre d'exemple si on considère un graphe quelconque  $G$ , et une numérotation des sommets telle que  $v(1)$  est le premier,  $v(2)$  le second, etc., alors  $G^{-1}$  possède un seul sommet  $v(1)$ , un ensemble frontière  $F_1 = \{v(1)\}$  et une seule solution partielle  $[v(1)]$  de valeur 1 (une seule couleur pour colorier  $v(1)$  est nécessaire). Les solutions partielles de  $G^{-2}$  seront obtenues à partir de  $[v(1)]$  comme suit :

- Si  $(v(1), v(2)) \in E$ 
  - Si  $\exists(v(1), v(k)) \in E, (k > 2)$  alors  $F_2 = \{v(1), v(2)\}$  et la seule solution partielle est  $[v(1)][v(2)]$  de valeur associée 2.

Noeud	Solution Partielle	Evaluation
$N_5$	[cf]	$\infty$
	[c][f]	2
$N_4$	[abc]	$\infty$
	[ac][b]	$\infty$
	[a][bc]	2
	[c][ab]	$\infty$
	[a][b][c]	3
$N_2$	[bcd]	$\infty$
	[bc][d]	$\min(\max(\infty, \infty, 2), \max(\infty, 2, 2), \max(2, \infty, 2), \max(2, 2, 2)) = 2$
	[b][cd]	$\infty$
	[c][bd]	$\infty$
	[b][c][d]	$\min(\max(\infty, \infty, 3), \max(\infty, \infty, 3), \max(\infty, 3, 3), \max(2, \infty, 3), \max(2, \infty, 3), \max(2, 3, 3)) = 3$
$N_3$	[dg]	$\infty$
	[d][g]	2
$N_1$	[cde]	$\infty$
	[c][de]	$\infty$
	[d][ce]	$\infty$
	[e][cd]	$\infty$
	[c][d][e]	$\min(\max(2, \infty, 3), \max(2, 2, 3), \max(\infty, \infty, 3), \max(\infty, 2, 3), \max(3, \infty, 3), \max(3, 2, 3)) = 3$
$\chi(G) = \min(\infty, \infty, \infty, \infty, 3) = 3$		

FIGURE 2.2 – Génération et évaluation des solutions partielles suivant la décomposition arborescente de la figure 2.1

- Sinon  $F_2 = \{v(2)\}$  et la seule solution partielle est  $[v(2)]$  de valeur associée 2.
- Sinon
  - Si  $\exists(v(1), v(k)) \in E$ , ( $k > 2$ ) alors  $F_2 = \{v(1), v(2)\}$  et les solutions partielles sont  $[v(1)][v(2)]$  de valeur associée 2 et  $[v(1) v(2)]$  de valeur associée 1.
  - Sinon  $F_2 = \{v(2)\}$  et la seule solution partielle est  $[v(2)]$  de valeur associée 1. On ne garde en mémoire que la meilleure des solutions de  $G^{-2}$  relativement à  $F_2$ .

Après  $n$  itérations, le nombre chromatique  $\chi(G)$  est la valeur minimale parmi l'ensemble des valeurs associées aux solutions partielles de  $G^{-|V|}$ .

Lors de chaque itération le nombre de solutions partielles considérées est donc relatif au nombre de partitions de l'ensemble frontière  $F_i$ , c'est à dire exponentiel en fonction du nombre de sommets de  $F_i$  et non du nombre de sommets de  $G^{-i}$ . Néanmoins, ce nombre peut être prohibitif si la taille de l'ensemble frontière est trop grand. La complexité générale de la méthode de décomposition-linéaire est donc en  $O(n.e^{F_{max}})$ .

Nous avons donc apporté deux améliorations à la version de base de l'algorithme, la première concerne la renumérotation des sommets et la seconde une approche dichotomique de résolution.

Trouver la numérotation optimale des sommets donnant la cardinalité minimale pour  $F_{max}$  est un problème NP-Complet. Afin de déterminer une numérotation tendant à minimiser cette valeur, nous avons utilisé le principe du double parcours en largeur du graphe ainsi que la numérotation

par clique. Néanmoins, pour beaucoup d'instances,  $|F_{max}|$  reste élevée et donc le nombre de solutions partielles à considérer trop grand. Or, si au lieu du problème de coloration, nous cherchions à résoudre un problème de  $k$ -coloration, il deviendrait inopportun de considérer les partitions de plus de  $k$  blocs, ce qui réduit le nombre de solutions partielles à considérer lors de chaque itération. A partir de cette remarque, nous avons proposé un algorithme dichotomique permettant de résoudre le problème de coloration par la résolution de plusieurs problèmes de  $k$ -coloration.

Considérer le problème de  $k$ -coloration nous a également ouvert de nouvelles perspectives de pré-traitements du graphe en amont, afin d'en réduire au maximum la taille. Ces pré-traitements sont succinctement expliqués dans la section suivante.

### 2.1.5 Pré-traitements

Il est ici question de pré-traitements ayant pour but de réduire la difficulté d'un problème de  $k$ -coloration en réduisant la taille du graphe par suppression des sommets qui n'ont pas un rôle majeur dans la détermination du nombre chromatique, ou à contraindre celui-ci par l'ajout de certaines arêtes. L'ensemble des opérateurs de réduction garantit que si le graphe réduit est  $k$ -coloriable, alors le graphe initial est également  $k$ -coloriable.

#### Réduction par voisinage

Etant donné un graphe  $G$ , pour chaque couple de sommets  $(x, y) \in V \times V$  tels que  $(x, y) \notin E$ , alors si le voisinage  $\Gamma(y)$  de  $y$  est inclu dans le voisinage  $\Gamma(x)$  de  $x$ ,  $y$  et ses arêtes adjacentes peuvent être supprimés du graphe. En effet, supposons que  $k - 1$  couleurs sont nécessaires pour colorier les voisins de  $x$ . Le sommet  $x$  peut alors prendre la  $k$ -ième couleur. Les sommets  $x$  et  $y$  ne sont pas voisins et les voisins de  $y$  sont déjà coloriés avec au plus  $k - 1$  couleurs. Si  $G \setminus \{y\}$  est  $k$ -coloriable alors  $G$  est  $k$ -coloriable également. La complexité en temps de cette réduction est de l'ordre de  $O(m \cdot \Delta(G))$ , bornée supérieurement par  $O(n^3)$ .

#### Réduction par degré

Pour chaque sommet  $x$ , si le degré de  $x$  est strictement inférieur à  $k$ ,  $x$  et ses arêtes peuvent être supprimés du graphe. Supposons que  $x$  ait  $k - 1$  voisins. Dans le pire des cas, ces voisins doivent tous avoir des couleurs différentes. Alors le sommet  $x$  peut prendre la  $k$ -ième couleur. L'affectation de cette couleur à  $x$  s'interférera pas dans la coloration du reste du graphe puisque tous les voisins de  $x$  sont déjà coloriés. La complexité en temps de cette réduction est  $O(n)$ .

#### Fusion de sommets

Soit une clique  $C$  du graphe  $G = (V, E)$ , de taille  $c$ . Pour chaque couple de sommets  $(x, y) \in V \times V$  tels que  $(x, y) \notin E$ ,  $x \notin C$  et  $y \in C$ , si  $x$  est adjacent à tous les sommets de  $C \setminus \{y\}$ , alors  $x$  et  $y$  peuvent être fusionnés de la façon suivante : chaque voisin de  $x$  devient un voisin de  $y$ , puis  $x$  et ses arêtes adjacentes sont supprimés du graphe. La complexité en temps de ce pré-traitement est  $O(n \cdot c)$ .

#### Ajout d'arêtes

Soit une clique  $C$  du graphe  $G = (V, E)$ , de taille  $c$ . Pour chaque couple de sommets  $(x, y) \in V \times V$  tels que  $(x, y) \notin E$ , si pour tout  $z \in C$  nous avons  $(x, z) \in E$  ou  $(y, z) \in E$ , alors l'arête  $(x, y)$  peut être ajoutée au graphe.  $x$  doit nécessairement prendre une couleur parmi les couleurs de  $C \setminus \Gamma(x)$ . Puisque  $\Gamma(y) \supseteq C \setminus \Gamma(x)$ ,  $c(x) \neq c(y)$ . Cette contrainte peut être représentée par une arête entre  $x$  et  $y$ . La complexité en temps de ce pré-traitement est  $O(m \cdot c)$ , bornée supérieurement par  $O(n^2 \cdot c)$ .

Ces opérateurs de réduction sont appliqués de manière itérative, jusqu'à ce que plus aucun d'entre eux ne soit éligible.

### 2.1.6 Synthèse des résultats sur la coloration

Les travaux autour de la thèse de Florence Mendes, concernaient principalement la résolution optimale du problème du nombre chromatique d'un graphe. L'algorithme LDC proposé, regrou-

pant tous les principes ci-dessus mentionnés, fut comparé aux résultats de la littérature tel que l'algorithme de branch-and-cut de Diaz & al. [54] et celle de Herrmann & Hertz [55] basée sur la détermination de sous-graphes critiques. L'ensemble des algorithmes et des résultats sont présentés dans la thèse de Florence Mendes.

Les pré-traitements développés ont été appliqués sur deux types de benchmarks : un ensemble de graphes aléatoirement générés dont les caractéristiques retenues sont le nombre de sommets et la densité du graphe ; un ensemble de graphes de la littérature, issus des célèbres benchmarks SGB, COLOR02 et DIMACS.

Nous n'avons obtenu que de faibles réductions de taille sur les graphes aléatoires et spécialement pour les graphes de densité 0.5 pour lesquels aucun sommet n'est supprimé (0%). A contrario, pour certaines familles de graphes de SGB, comme les graphes *miles*, *zero*, et *book graphs* la réduction est totale (100%). En moyenne sur les benchmarks de la littérature la réduction est de 62,43 %. Mais une réduction de  $x\%$  de la taille du graphe n'implique pas une réduction de  $x\%$  de la taille maximale des ensembles frontières. Comme exemple je citerais l'instance *fpsol2.i.2* pour laquelle  $|V| = 451$  et  $|F_{max}| = 50$ , qui est réduit à 80% de sa taille ( $|V^*| = 87$ ), mais seulement à 10% de la taille maximum de ses ensembles frontières ( $|F_{max}^*| = 45$ ).

L'algorithme LDC sur les graphes aléatoires n'est pas performant comparés aux autres méthodes de la littérature, notamment pour les graphes de densité 0.5. Ce type de graphe implique un  $F_{max}$  de grande taille, et trop peu de contraintes pour limiter le nombre de solutions partielles. Les résultats sont plus compétitifs sur les benchmarks de la littérature. Les bornes et les nombres chromatiques fournis par LDC sont comparables en précision aux méthodes de la littérature mais les meilleurs solutions sont atteintes plus rapidement (30 minutes maximum, contre 120 minutes pour les autres méthodes). Une instance a été résolue de manière optimale pour la première fois (4-Insertion-3). La cardinalité maximale des  $F_{max}$  des graphes traités est d'au plus 30. Globalement, LDC se comporte efficacement pour les graphes de petite largeur linéaire ( $F_{max} \leq 10$ ) car dans ce cas le nombre de partitions n'est pas trop élevé, et pour les graphes de largeur linéaire supérieure ( $15 \leq F_{max} \leq 46$ ) mais de forte densité. En effet, pour ces dernières le nombre de partitions valides des ensembles frontières est restreint.

## 2.2 Somme chromatique et force d'un graphe

### 2.2.1 Introduction

A la suite de nos travaux sur le problème du nombre chromatique, nous nous sommes intéressés à un problème connexe, peu étudié de manière expérimentale dans la littérature, le problème de la somme chromatique d'un graphe. Etant donnée la difficulté à borner le nombre de couleurs pour ce problème, la résolution par une décomposition linéaire du graphe ne nous a pas parue appropriée. Nous avons donc dans un premier temps, abordé ce problème dans le cadre du projet "Somme COloration et aPplications (SCOOP)" soutenu par le Conseil Régional de Picardie. Ce projet fut l'objet d'une collaboration entre le laboratoire HeuDiaSyC de Université de Compiègne (Pr Aziz Moukrim) et le laboratoire MIS de l'UPJV (Yu Li et Corinne Lucet ) et a permis le co-encadrement de la thèse de Kaoutar Sghiouer (UTC, 2007-2011). Nous y avons développé des algorithmes gloutons permettant la construction de solutions valides, puis des heuristiques (méthode de recherche locale de type destruction/construction) et enfin un algorithme mémétique dédié à la résolution du problème de la somme chromatique d'un graphe. Ces travaux ont donné lieu à des publications en conférence internationales [69, 70, 71] et nationales (ROADEF 2009, 2010, 2011). Kaoutar Sghiouer est actuellement responsable technique Data Science dans la société BULL-Atos Technologies à Paris.

Ensuite, et en collaboration avec le Pr. Chumin Li (MIS/UPJV), nous nous sommes intéressés à au développement de méthodes exactes pour résoudre ce problème. Ces travaux ont été menés à travers l'encadrement de la thèse de Clément Lecat (UPJV, 2014-2017), qui a bénéficié d'une allocation ministérielle. Tout d'abord nous avons modélisé le problème sous la forme des problèmes Weighted Partial MaxSAT et Weighted Partial MinSAT afin de tester des solveurs existants pour la résolution de MSCP. Nous avons également développé deux algorithmes de type branch-and-

bound BBMSCP et 3LMSCP qui ne seront pas exposés dans ce document, mais dont le détail se trouve dans la thèse de Clément Lecat. Enfin par une analyse de l'espace des solutions, nous avons proposé une modélisation des solutions sous forme de séquences d'entiers, appelés *motifs*. A partir de ceux-ci nous avons dégagé deux bornes supérieures de la force du graphe  $UB_A$  et  $UB_S$  ainsi qu'une borne inférieure de la somme chromatique  $LB_\Sigma$ . Les résultats numériques obtenus furent très convaincants. Ces travaux ont été publiés dans la revue internationale *Discrete Applied Mathematics* [7], dans des conférences internationales dont AAAI 2017 [8] et CP 2015 [72] ainsi que dans des conférences nationales (ROADEF 2015 et 2017 ; JFPC 2015, 2017 et 2018). Clément Lecat a obtenu le prix du jeune chercheur de la ROADEF en 2017 pour la présentation des travaux sur la réduction de l'espace des solutions par motifs. Il est actuellement Ingénieur R&D IA chez EVOLUCARE Technologies.

### 2.2.2 Définition du problème

Le problème de la somme coloration minimum d'un graphe est un cas particulier du problème de partition chromatique de coût optimal (OCCP). Ce problème de coloration fut introduit par Supowit en 1987 [73]. Sachant qu'à chaque couleur est associé un poids, l'objectif du problème OCCP est donc de déterminer une coloration valide respectant les contraintes de voisinage, et minimisant la somme des poids des couleurs utilisées.

Comme pour le nombre chromatique, les applications relatives à OCCP se trouvent principalement dans les problèmes d'ordonnancement ou d'allocation de ressources. Le poids d'une couleur peut par exemple représenter une date dans le cas d'un ordonnancement de processus. Minimiser le nombre des couleurs revient dans ce cas à minimiser le temps global d'achèvement des processus, alors que pour le problème de la somme chromatique il correspond à la minimisation du temps moyen d'achèvement et s'apparente en cela à une mesure de qualité de service.

**Définition 21 (OCCP)** Soit  $G = (V, E)$  un graphe et  $\{c_1, c_2, \dots, c_k\}$  un ensemble de couleurs auquel est associé un ensemble de poids  $\{w_1, w_2, \dots, w_k\}$  tel que le poids  $w_i$  est associé à la couleur  $c_i$ . Le problème de la partition chromatique de coût minimal d'un graphe  $G$ , notée  $\sum_{OCCP}(G)$ , consiste à déterminer une coloration valide  $X$  de  $G$ , pour laquelle la somme des poids est minimale, comme défini ci-dessous :

$$\sum_{OCCP}(G) = \min\{\sum_{i=1}^{|X|} w_i \mid X_i \mid, X \text{ est une coloration valide de } G\}$$

Le problème de la somme coloration minimum d'un graphe (MSCP), introduit par Kubicka et Schwenk en 1989 [74] est le cas particulier où chaque couleur est représentée par un entier  $i$ , dont le poids associé est la valeur de cette couleur, i.e. à la couleur  $i$  est associé le poids  $w_i = i$ .

Comme pour OCCP, l'objectif est de minimiser la somme des poids des couleurs (voir définition 22). Il a été montré que MSCP est un problème NP-difficile [74].

**Définition 22 (MSCP)** Soit  $G = (V, E)$  un graphe. Le problème de la somme chromatique de  $G$ , notée  $\sum(G)$ , consiste à déterminer une coloration valide  $X$  de  $G$ , pour laquelle la somme des poids est minimale, comme définie ci-dessous :

$$\sum(G) = \min\{\sum_{i=1}^{|X|} i \mid X_i \mid, X \text{ est une coloration valide de } G\}$$

Le nombre minimum de couleurs utilisées dans une solution optimale de MSCP est appelé la *force* du graphe, notée  $s(G)$  (voir définition 23). La particularité de MSCP est que cette force peut-être arbitrairement plus grande que le nombre chromatique du graphe. L'exemple de la figure 2.3 illustre la différence entre le problème du nombre chromatique et le problème de la somme chromatique. En effet, il est toujours possible de colorier un arbre avec deux couleurs ( $\chi(G) = 2$ ), mais sur cet exemple il est nécessaire d'utiliser une troisième couleur afin de minimiser la somme ( $s(G) = 3$  et  $\sum(G) = 11$ ). La force d'un arbre est généralement relative à son degré, plutôt qu'à son nombre chromatique. L'espace des solutions pour MSCP est donc difficile à explorer, car il est difficile de borner le nombre de couleurs à considérer dans la recherche d'une solution optimale.

**Définition 23 (Force du graphe)** Soit  $G = (V, E)$  un graphe. La force du graphe  $G$ , notée  $s(G)$  se définit comme suit :

$$s(G) = \min\{|X|, \text{ telle que } \sum_{i=1}^{|X|} i \cdot |X_i| = \sum(G)\}$$

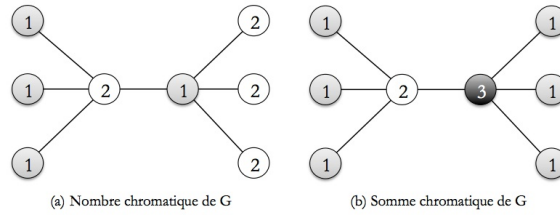


FIGURE 2.3 – Nombre chromatique vs Somme chromatique :  $\chi(G) = 2$ ;  $\sum(G) = 11$ ;  $s(G) = 3$

Comme nous l'avons souligné dans l'introduction, une coloration valide d'un graphe est un ensemble de classes couleurs tel que chaque classe couleur forme un stable. La permutation de deux classes couleurs d'une coloration n'impacte donc pas la validité de cette dernière, mais a des conséquences sur la somme coloration associée. Par conséquent, étant donné une coloration  $X$ , il existe un ensemble de colorations symétriques  $\phi(X)$ , parmi lesquelles on distinguera ce que nous appelons *les colorations majeures*, précisément décrites par la définition 24. Ces colorations majeures sont de somme coloration minimale dans  $\phi(X)$ .

**Définition 24 (Coloration Majeure)** Soit  $G = (V, E)$  un graphe et  $X$  une coloration valide de  $G$ . L'ensemble  $\phi(X)$  est l'ensemble des permutations des classes couleurs de la coloration  $X$ . Toute coloration  $X' \in \phi(X)$  telle que  $X' = \{X'_1, X'_2, \dots, X'_k\}$  avec  $|X'_1| \leq |X'_2| \leq \dots \leq |X'_k|$  est appelée coloration majeure et vérifie la propriété :  $\forall X'' \in \Phi(X), \sum(X'') \geq \sum(X')$

Chaque coloration de  $\phi(X)$  peut être ramenée à une coloration majeure par un simple tri des classes couleurs suivant leur cardinalité. Dans l'ensemble des méthodes de résolution, la solution proposée est toujours ramenée à une coloration majeure.

Les différentes méthodes de résolution dédiées à la résolution du problème de la somme chromatique sont classifiables en deux catégories. La première catégorie regroupe les approches dites exhaustives ou exactes qui calculent la solution optimale, et la seconde catégorie regroupe les approches dites non-exhaustives ou incomplètes qui fournissent une solution valide sans garantie d'optimalité. L'intérêt majeur des méthodes incomplètes est leur temps de résolution. Dans la littérature, ces méthodes sont principalement représentées par des méthodes gloutonnes telles que EXSCOL [75], MDSAT et MRLF [69], des méthodes à base de recherche locale telle que BLS [76], MDS(5)+LS [77], IDCA [71], et des algorithmes évolutionnaires tels que MA-MSCP [71, 70], MASC [78], et HESA [79].

### 2.2.3 Méthodes approchées

Nos premières contributions pour ce problème se situent dans la famille des méthodes incomplètes. Nous avons proposé deux algorithmes gloutons, MRLF et MDAT, un algorithme de recherche locale basé sur le principe de Destruction/Construction IDCA et enfin un algorithme de type évolutionnaire MA-MSCP.

#### Algorithmes glouton MDSATUR et MRLF

Nous avons dans un premier temps proposé deux algorithmes gloutons MDSATUR et MRLF, basés sur les algorithmes gloutons de référence en coloration de graphe, DSATUR de Daniel Brélaz ?? et RLF de Fred Glover [80].

La présentation générale d'un algorithme glouton comprend quatre éléments de base : une fonction objectif  $f$ , une solution partielle  $S$  (un sous ensemble de sommets coloriés éventuellement vide), un ensemble de candidats  $L$  (un sous-ensemble de sommets non coloriés) et une règle de sélection  $r$ . A chaque itération un candidat est choisi suivant la règle  $r$ , il sort de  $L$ , une couleur valide lui est attribuée, il entre en  $S$ , et ainsi de suite jusqu'à ce que tous les sommets soient dans  $S$ . Construire de nouveaux algorithmes gloutons comme MRLF et MDSAT pour résoudre MSCP, revient donc à proposer une nouvelle règle de sélection  $r$  tendant à minimiser la somme des couleurs.

**MDSAT** DSATUR est un célèbre algorithme glouton dont les règles et les critères de sélection utilisés sont  $dsat(v)$  et  $d\Gamma^-(v)$ .

Soit  $v$  un sommet non colorié et  $\Gamma(v)$  l'ensemble de ses sommets voisins. Cet ensemble peut être divisé en deux sous ensembles. Le premier contient les voisins de  $v$  déjà coloriés, noté par  $\Gamma^+(v)$  et le deuxième ceux qui ne le sont pas encore, noté par  $\Gamma^-(v)$ .  $\Gamma(v) = \Gamma^+(v) \cup \Gamma^-(v)$ . Le nombre de couleurs différentes utilisées dans  $\Gamma^+(v)$  est noté par  $dsat(v)$ , et appelé le degré de saturation du sommet  $v$  [Br9]. La cardinalité de  $\Gamma^-(v)$  est notée par  $d\Gamma^-(v)$ . La règle de sélection de DSATUR est représentée par l'expression suivante :

$$[max(dsat(v)); max(d\Gamma^-(v))]$$

Le sommet ainsi sélectionné est colorié avec sa plus petite couleur disponible  $c_{min}(v)$ , sans mesurer l'impact de ce choix sur la structure de la solution. Nous avons donc introduit le critère  $\Gamma_{dsat}^-(v)$  qui désigne l'ensemble des voisins non coloriés de  $v$  ayant  $c_{min}(v)$  comme couleur disponible, et  $d_{\Gamma_{dsat}^-}(v)$  sa cardinalité. Puis un second critère  $\Gamma_{dsatNC}^-(v)$  qui désigne l'ensemble des voisins non coloriés de  $v$  n'ayant pas  $c_{min}(v)$  comme couleur disponible et  $d_{\Gamma_{dsatNC}^-}(v)$  sa cardinalité.  $d_{\Gamma_{dsat}^-}(v)$  et  $d_{\Gamma_{dsatNC}^-}(v)$  sont utilisés pour mesurer l'impact de la coloration d'un sommet  $v$ . Le critère de choix est représenté par l'expression suivante :

$$[min(d_{\Gamma_{dsat}^-}(v)/d_{\Gamma_{dsatNC}^-}(v))]$$

**MRLF** Dans l'article d'origine de Leighton, RLF est interprété comme un algorithme de plus grand stable d'abord, comme son nom l'indique *Recursive Largest First*. Le principe de RLF est le suivant : pour toute couleur  $i = 1, \dots, k$ , il construit étape par étape la classe de couleur  $X_i$  en choisissant le sommet suivant une règle de sélection et lui attribut sa plus petite couleur disponible  $c_{min}(v)$ . Contrairement à DSATUR, RLF évalue l'impact de la coloration du sommet élu au sens stricte par l'emploi des critères  $d\Gamma_{c_{min}C}^-(v)$  et  $d\Gamma_{c_{min}NC}^-(v)$  qui représentent respectivement le nombre de voisins non coloriés partageant le même  $c_{min}()$  et le nombre de voisins non coloriés ne partageant pas le même  $c_{min}()$ . La règle de sélection est donnée par l'expression suivante :

$$[max(d_{\Gamma_{c_{min}C}^-}(v)); min(d_{\Gamma_{c_{min}NC}^-}(v))]$$

L'analyse des critères de RLF, relativement à MSCP nous a conduit à les réemployer afin de simultanément maximiser la taille des ensembles indépendants (les classes couleurs) et à en minimiser le nombre de ces ensembles. Deux caractéristiques qui interviennent dans la minimisation de somme des couleurs. Nous avons proposé deux critères de choix différents, donnant les deux algorithmes gloutons suivants :

$$RLF_{Inv} : [min(d_{\Gamma_{c_{min}C}^-}(v)); max(d_{\Gamma_{c_{min}NC}^-}(v))]$$

$$MRLF : [min(d_{\Gamma_{c_{min}C}^-}(v))/d_{\Gamma_{c_{min}NC}^-}(v)]$$

Les tests effectués sur 76 instances de DIMACS montrent l'intérêt de l'analyse de la structure des solutions du problème dans l'élaboration de méthodes constructives. Ces résultats sont synthétisés dans le tableau 2.4, qui pour chaque méthode donne le nombre de fois ou la meilleure des solutions donnée par ces gloutons a été atteinte parmi les 76 instances.

	<i>DSATUR</i>	<i>RLF</i>	<i>RLF<sub>inv</sub></i>	<i>MRLF</i>	<i>MDSAT</i>
nbBest	1	13	39	33	33

FIGURE 2.4 – Comparaison des algorithmes gloutons pour MSCP

### Destruction/Construction IDCA

Afin d'améliorer les solutions obtenues par les algorithmes gloutons MDSAT et MRLF, nous avons proposé l'heuristique IDCA, consistant à détruire une faible partie d'une solution puis à reconstruire suivant des critères d'optimalité locale. Cette méthode est inspirée des travaux de Ruiz et Stützle ?? à laquelle nous avons intégré un mécanisme de diversification spécifique au problème de la somme chromatique.

**Destruction** La solution courante est donc représentée par une coloration  $X = \{X_1, X_2, \dots, X_k\}$ . La phase de destruction consiste à extraire  $D$  sommets de leur classe de couleur respective. Ces sommets sont choisis aléatoirement.

**Construction** Les sommets extraits sont ordonnés de manière aléatoire. Soit  $v_\alpha$  un de ces sommets extrait de la classe couleur  $X_i$ . Alors si  $\exists X_j, |X_i| \leq |X_j|$  et  $j$  est une couleur disponible pour le sommet  $v_\alpha$ , alors celui-ci est inséré dans la classe  $X_j$ , sinon il retourne dans sa classe initiale  $X_i$ .

On notera que ce processus de Destruction/Construction est strictement améliorant, d'où la nécessité d'y apporter de la diversification, tout simplement en réaffectant une nouvelle couleur disponible (et donc potentiellement non améliorante) à un nombre  $D'$  de sommets choisi aléatoirement. Ces deux phases sont alternativement appliquées à la solution courante pendant un nombre  $iter_{max}$  prédéfini d'itérations. La complexité de l'algorithme IDCA est en  $O((n.k + m).iter_{max})$  où  $n$  est le nombre de sommets,  $m$  le nombre d'arêtes,  $k$  le nombre de couleurs et  $iter_{max}$  le critère d'arrêt de la recherche locale.

IDCA est un algorithme de recherche locale simple, qui fut combiné comme opérateur de mutation à l'algorithme mémétique ci-après exposé.

### Algorithme mémétique MA-MSCP

Les algorithmes mémétiques (MA) sont des algorithmes évolutionnaires hybrides constitués de deux phases d'évolution : une phase d'évolution génétique et une phase d'évolution locale. Ce type de métaheuristique a été proposé par Moscato & al. [81] pour des problèmes combinatoires sur lesquels la seule sélection génétique n'est pas suffisamment efficace. Comme tout algorithme évolutionnaire, un MA fait évoluer de manière itérative une population d'individus (des solutions) par les principes génétiques de croisement, sélection et mutation, puis pour chacun des individus de la population il applique un processus d'amélioration locale. Les différents travaux de la littérature ont montré de bonnes performances sur le problème du nombre chromatique d'un graphe [82] ainsi que pour des problèmes d'ordonnancement [83] et de tournées [84].

Quelques soient les problèmes traités, les principales phases d'un algorithme mémétique sont représentées par l'algorithme 1.

La principale différence entre un algorithme génétique classique et un algorithme mémétique est donc représenté par la ligne 7 de l'algorithme, où chaque nouvel individu obtenu par croisement est amélioré par un ou plusieurs opérateurs, accomplissant une recherche locale. Dans l'algorithme mémétique MA-MSCP, développé durant la thèse de Kaoutar Sghiouer, ces principales phases sont reprises, à l'exception de l'application de la recherche locale pour chacun des enfants. Le cout associé au temps de calcul étant trop élevé, nous avons choisi d'utiliser cette recherche locale comme opérateur de mutation. Le réglage de l'ensemble des paramètres à fait l'objet d'études statistiques reportées dans la thèse de Kaoutar Sghiouer.

**Codage et Population initiale** Chaque individu de la population est une coloration valide, pour laquelle il n'y a donc pas de conflits à gérer entre les sommets, représenté par la liste des



**Algorithm 1** Algorithme Mémétique**Entrées :** Une fonction fitness  $f$ **Sorties :**  $X^*$  la meilleure solution

- 1: Génération d'une population initiale  $P$
- 2: Evaluation des individus de  $P$  par  $f$
- 3:  $X^* \leftarrow$  meilleur individu parmi  $P$
- 4: **tantque** La condition d'arrêt n'est pas vérifiée **faire**
- 5:   Sélection d'un sous-ensemble d'individus (les parents)
- 6:   Croisements des parents pour générer de nouveau individus (les enfants)
- 7:   Application d'un **opérateur d'amélioration locale** pour chaque enfant
- 8:   Mutation appliquée à un sous-ensemble des enfants
- 9:   Mise à jour de la population  $P$  en ne gardant que les meilleurs enfants relativement à  $f$
- 10:    $X \leftarrow$  meilleure solution de  $P$
- 11:   **si**  $f(X) \leq f(X^*)$  **alors**
- 12:      $X^* \leftarrow X$
- 13:   **fin si**
- 14: **fin tantque**
- 15: **retourner**  $X^*$

classes couleurs. Garder l'information de cette structure en classes couleurs dans le codage des solutions est une donnée importante pour le problème de la somme coloration, dont la fonction objectif tend à privilégier les classes de grande taille. Pour la génération de la population initiale, nous avons utilisé l'algorithme IDCA, pour 25% des individus et une génération aléatoire pour les 75% restant. La taille de la population est fixée à 20.

**Fonction fitness** La fonction fitness utilisée pour chaque individu  $X$  calcule la somme des couleurs de la coloration majeure associée à  $X$ . Les classes couleurs sont simplement ordonnées suivant leur cardinalité.

**Sélection des parents** Pour la sélection des parents, nous avons utilisé le tournoi binaire, souvent donné comme le plus efficace dans la littérature, ce qui fut confirmé pour notre problème.

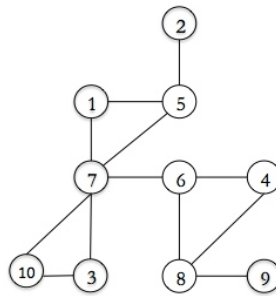


FIGURE 2.5 – Un graphe

**Croisement** L'opérateur de croisement, *cross-MSCP*, que nous avons proposé est inspiré de celui de Galinier et Hao [85] pour le problème du nombre chromatique. Adapté au problème MSCP, *cross-MSCP* a pour objectif de maximiser la taille des classes couleurs des enfants engendrés. Cette caractéristique impacte évidemment la somme des couleurs, les couleurs de poids faible étant affectées aux classes de grande taille par la fonction fitness. *Cross-MSCP* construit un enfant à partir des plus grandes classes couleurs de chacun de ses deux parents, en gardant la coloration valide. Le schéma de la figure 2.6 illustre les étapes de *cross-MSCP* sur le graphe de la figure 2.5.

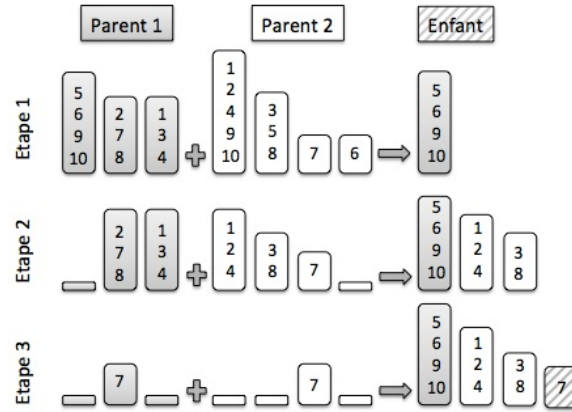


FIGURE 2.6 – Les étapes d'un croisement par cross-MSCP

**Opérateur d'amélioration** Nous avons utilisé l'heuristique IDCA comme opérateur d'amélioration des individus. IDCA est appliqué comme mutation suivant une certaine probabilité  $pm$  qui évolue de manière dynamique relativement au nombre de mises à jour de la meilleure solution au cours du processus. Lors de la phase de destruction, IDCA extrait  $D$  sommets des classes couleurs de l'individu.  $D$  représente donc le taux de perturbation. La valeur assignée à ce paramètre fût calculé en minimisant la variance de l'écart relatif à la meilleure solution, sur un ensemble de benchmarks difficiles, issue de la famille des instances aléatoires  $dsjcX.Y$  de DIMACS. La valeur  $D = 8$  fût retenue.

**Résultats** Les algorithmes MRLF, DSATUR, IDCA et MA-MSCP sont tous dédiés à MSCP. Ils ont été évalués sur un grand nombre d'instances de DIMACS et COLOR02, afin de les comparer aux meilleurs résultats de la littérature du moment [86]. Les résultats obtenus montraient l'intérêt d'une telle approche, par l'amélioration stricte des scores, c'est à dire les bornes supérieures de la somme chromatique, des instances de la littérature. Certaines instances ont été résolues de manière optimale, par l'utilisation de MA-MSCP sur le graphe complémentaire, qui fournit comme mentionné dans la paragraphe D suivant, une borne Inférieure de la somme chromatique.

### Bornes Inférieures pour MSCP et le problème PCMSCP

La littérature ne présentait à notre connaissance qu'une seule borne inférieure non naïve pour la somme chromatique des graphes, il s'agit de la borne  $LB_{kok}$  de Kokosinski & al. [86] présentée par l'équation 2.1. Les résultats obtenus par celle-ci sont très loin de la somme chromatique de presque la totalité des benchmarks.

$$LB_{kok}(G) = (|V| - \chi(G)) \cdot \frac{\chi(G) \cdot (\chi(G) + 1)}{2} \quad (2.1)$$

Or, si on considère un graphe partiel  $G' = (V, E')$  de  $G = (V, E)$ , toute coloration valide de  $G$  est une coloration valide de  $G'$ . A fortiori, la solution optimale  $X_{opt}$  de  $G$  est aussi une solution valide de  $G'$ , de quoi on peut déduire la relation suivante :  $\Sigma(G) \geq \Sigma(G')$ . Donc, si il est possible de construire un graphe partiel de  $G$  pour lequel on sait calculer la somme chromatique de manière efficace, alors celle-ci est une borne inférieure de  $\Sigma(G)$ . Or, une partition en cliques de  $G$  représente un tel graphe partiel.

En effet, si on considère une clique  $C_i$  de taille  $n_i$ , alors sa somme chromatique est directement calculée par :  $\Sigma(C_i) = \frac{n_i \cdot (n_i + 1)}{2}$ . Ceci correspond à l'affectation des couleurs  $1, 2, \dots, n_i$  aux sommets de  $C_i$ . Ce qui implique directement la propriété 1.

**Propriété 1** Soit une partition du graphe  $G = (V, E)$  en  $k$  cliques  $C_1, C_2, \dots, C_k$ , telles que  $\forall i, j C_i \cap C_j = \emptyset$ . Alors, la somme chromatique du graphe partiel  $G'$  formé par l'union de ces cliques est :

$$\Sigma(G') = \Sigma(C_1) + \dots + \Sigma(C_k) = \sum_{i=1}^k \frac{n_i \cdot (n_i + 1)}{2}$$

A partir de cette propriété, nous avons défini un problème relatif à la détermination d'une partition d'un graphe en cliques, de manière à obtenir une borne inférieure de qualité pour la somme chromatique. Nous avons nommé ce problème *Partition en Cliques pour MSCP* (PCMSCP) (voir définition 25).

**Définition 25 (PCMSCP)** Soit  $G = (V, E)$  un graphe, le problème de *Partition en Cliques pour MSCP* consiste à trouver une partition de  $V$  en sous-ensembles disjoints  $V_1, V_2, \dots, V_k$ , telle que  $\forall i = 1, 2, \dots, k$  le graphe induit par  $V_i$  est une clique et  $\sum_{i=1}^k \frac{n_i \cdot (n_i + 1)}{2}$  est maximum. La solution optimale de PCMSCP est notée  $\sum_{clique}(G)$ .

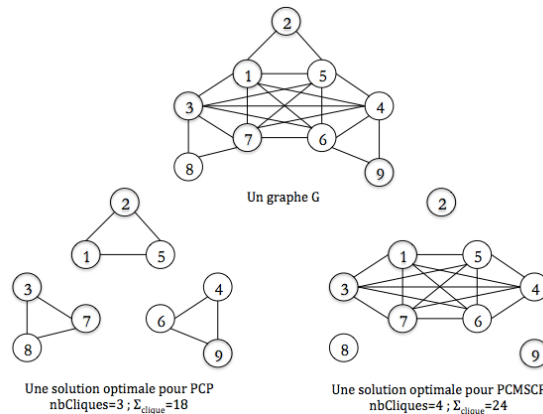


FIGURE 2.7 – PCP vs PCMSCP

PCMSCP est un problème proche du problème de partition en cliques d'un graphe (PCP), pour lequel l'objectif est de minimiser le nombre de cliques (dual du problème du nombre chromatique). Néanmoins, ces deux problèmes ne partagent pas toujours les mêmes solutions optimales comme l'illustre la figure 2.7. Nous avons démontré que PCMSCP comme PCP est un problème NP-difficile [5]. De plus, si  $\omega$  est la taille de la clique maximum de  $G$ , et  $r = n - \omega \lfloor \frac{n}{\omega} \rfloor$  alors  $\sum_{clique}(G)$  est supérieurement bornée par  $MAX_{LBclique}$  définie dans l'équation 2.2

$$MAX_{LBclique} = \frac{\omega \cdot (\omega + 1)}{2} \cdot \lfloor \frac{n}{\omega} \rfloor + \frac{r \cdot (r + 1)}{2} \quad (2.2)$$

Une première approche pour obtenir une partition en cliques du graphe, consiste à extraire des cliques une par une du graphe, en utilisant différents critères pour l'ordre d'exploration des sommets, comme le degré par exemple. Lors des travaux de thèse de Kaoutar Sghiouer, nous nous sommes basés sur la propriété 2 pour obtenir une borne inférieure du nombre chromatique grâce à l'algorithme MA-MSCP.

**Propriété 2** Soit un graphe  $G = (V, E)$ , on note  $\bar{G}(V, E)$  son complémentaire. Tout ensemble indépendant (stable) de  $G$  est une clique de  $\bar{G}$  et réciproquement. Donc, si on considère une coloration  $\bar{X} = X_1, X_2, \dots, X_k$  de  $\bar{G}$ , comme chaque classe de couleur  $X_i$  est un ensemble indépendant, il lui correspond une clique  $C_i$  dans  $G$ .  $\bar{X}$  définit donc une partition en clique de  $G$ .

Ainsi, l'utilisation de MA-MSCP sur le graphe complémentaire, nous permet d'obtenir une coloration de  $\bar{G}$ , donc une partition en cliques de  $G$  et par conséquent une borne inférieure du nombre chromatique, comme indiqué par l'algorithme 2. Nous avons comparé les deux approches et ainsi montré l'intérêt d'utiliser le graphe complémentaire avec MA-MSCP. Cette métaheuristique étant dédiée à MSCP, les solutions sont construites de façon à améliorer la somme des couleurs, comme l'illustre la figure 2.8.

**Algorithm 2** Algorithme Borne Inférieure pour MSCP**Entrées :** Un graphe  $G$ **Sorties :**  $LB$  une borne inférieure de MSCP pour  $G$ 

- 1:  $\bar{G} \leftarrow$  graphe complémentaire de  $G$
- 2: Coloration de  $\bar{G}$
- 3: Soit  $X_1, X_2, \dots, X_k$  les classes couleurs obtenues
- 4:  $LB \leftarrow \sum_{i=1}^k \frac{|X_i| \cdot (|X_i| + 1)}{2}$
- 5: **retourner**  $LB$

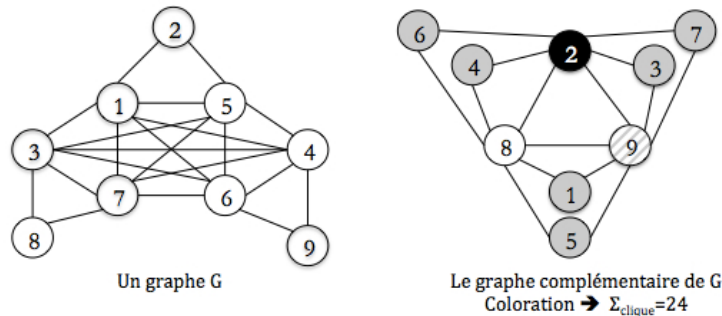


FIGURE 2.8 – Coloration du graphe complémentaire pour PCMSCP

**Synthèse des résultats des méthodes approchées pour MSCP**

Le problème de la somme chromatique était au début des travaux de thèse de Kaoutar Sghiouer en 2008 un problème peu étudié d'un point de vue expérimental. Seul Kokosinski & al avaient présenté en 2007 un algorithme génétique parallèle, testé sur un petit nombre d'instances (16) des célèbres benchmarks DIMACS et COLOR02.

Nous avons proposé pour le résoudre des algorithmes heuristiques : deux algorithmes gloutons MD-SAT et MRLF, un algorithme de recherche locale IDCA et un algorithme mémétique MA-MSCP. Ceux-ci furent testés sur un maximum d'instances (76) de la littérature. Les premiers résultats étaient de loin supérieurs à ceux de l'algorithme génétique parallèle, et montraient une somme coloration inférieure pour la totalité des 76 instances testées. Nous avons également formalisé un nouveau problème de partitionnement du graphe en clique PCMSCP, dont l'objectif est la maximisation de la somme chromatique du graphe partiel associé. Pour celui-ci nous avons proposé une borne supérieure algébrique. Les perspectives sont ouvertes pour PCMSCP, qui à notre connaissance n'a pas encore fait l'objet d'études expérimentales et/ou théoriques approfondies.

Depuis d'autres chercheurs se sont intéressés au problème de la somme chromatique [87, 88, 89] et ont améliorés certains de nos résultats par des approches de type métaheuristique. L'ensemble des algorithmes et des résultats sont présentés dans la thèse de Kaoutar Sghiouer ainsi que dans les articles [5, 69, 71]. La suite de nos travaux sur le problème de la somme chromatique adopte un tout autre angle, celui d'une approche de résolution exacte, à travers une formulation du problème de la somme chromatique basée sur un problème SAT, et d'une analyse de l'ensemble des solutions modélisé par un ensemble de « motifs ».

**2.2.4 Formulation SAT**

Le problème de satisfaisabilité booléenne (SAT) est un problème de décision dont l'objectif est de déterminer une instanciation d'un ensemble de variables booléennes, appelée interprétation afin de satisfaire une formule sous forme normale conjonctive (CNF). Par exemple la fonction  $F = (a \vee c \vee \neg d) \wedge (b \vee \neg c \vee d)$  est satisfaite par l'interprétation suivante : ( $a = vrai, b = vrai$ ). ( $a \vee c \vee \neg d$ ) est appelée une clause. Peuvent être formulés sous cette forme, les problèmes pour lesquels les solutions sont "oui" ou "non". C'est à dire les problèmes pour lesquels la question "existe-t-il une interprétation des variables qui satisfasse la fonction  $F$ ?" se pose. Ainsi pour le problème

du nombre chromatique (GCP) une formulation SAT correspondrait à répondre au problème de décision correspondant "Le graphe  $G$  est-il coloriable avec  $k$  couleurs?". La formulation SAT de ce problème fait intervenir les variables  $x_{ij}$  telles que  $x_{ij} = 1$  si le sommet  $i$  est colorié par la couleur  $j$ , et  $x_{ij} = 0$  sinon. Les contraintes liées à GCP sont représentées par trois types de clauses :

Type-A **tous les sommets doivent être coloriés** : pour chaque sommet  $i$ ,  $x_{i1} \vee x_{i2} \vee \dots \vee x_{ik}$  est une clause à satisfaire

Type-B **chaque sommet a exactement une couleur** : pour chaque sommet  $i$  et pour tout couple de couleurs  $(j_1, j_2) \in \{1, 2, \dots, k\}$ ,  $\neg x_{i,j_1} \vee \neg x_{i,j_2}$  est une clause à satisfaire

Type-C **deux sommets voisins sont de couleurs différentes** : pour chaque arête  $(i_1, i_2) \in E$  et pour toute couleur  $j \in \{1, 2, \dots, k\}$ ,  $\neg x_{i_1,j} \vee \neg x_{i_2,j}$  est une clause à satisfaire

Le tableau de la figure 2.10 liste l'ensemble des clauses dures de type A, B et C, pour le graphe de la figure 2.9 et pour une valeur de  $k=3$  couleurs.

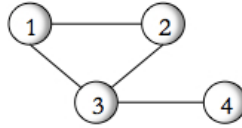


FIGURE 2.9 – Un graphe  $G=(V,E)$  de 4 sommets et de degré 3.

Clauses dures		
Type-A	Type-B	Type-C
$x_{11} \vee x_{12} \vee x_{13}$	$\neg x_{11} \vee \neg x_{12}$	$\neg x_{11} \vee \neg x_{21}$
$x_{21} \vee x_{22} \vee x_{23}$	$\neg x_{11} \vee \neg x_{13}$	$\neg x_{12} \vee \neg x_{22}$
$x_{31} \vee x_{32} \vee x_{33}$	$\neg x_{12} \vee \neg x_{13}$	$\neg x_{13} \vee \neg x_{23}$
$x_{41} \vee x_{42} \vee x_{43}$	$\neg x_{21} \vee \neg x_{22}$	$\neg x_{11} \vee \neg x_{31}$
	$\neg x_{21} \vee \neg x_{23}$	$\neg x_{12} \vee \neg x_{32}$
	$\neg x_{22} \vee \neg x_{23}$	$\neg x_{13} \vee \neg x_{33}$
	$\neg x_{31} \vee \neg x_{32}$	$\neg x_{21} \vee \neg x_{31}$
	$\neg x_{31} \vee \neg x_{33}$	$\neg x_{22} \vee \neg x_{32}$
	$\neg x_{32} \vee \neg x_{33}$	$\neg x_{23} \vee \neg x_{33}$
	$\neg x_{41} \vee \neg x_{42}$	$\neg x_{31} \vee \neg x_{41}$
	$\neg x_{41} \vee \neg x_{43}$	$\neg x_{32} \vee \neg x_{42}$
	$\neg x_{42} \vee \neg x_{43}$	$\neg x_{33} \vee \neg x_{43}$

FIGURE 2.10 – Ensemble des clauses dures pour GCP et MSCP sur le graphe de la figure 2.9

La formalisation des problèmes d'optimisation, comme MSCP, font référence à une généralisation du problème SAT, nommé *Weighted Partial MaxSAT* (WP-MaxSAT) qui introduit la notion de *clauses dures* et de *clauses souples pondérées*. L'objectif est alors de déterminer une interprétation satisfaisant l'ensemble des clauses dures (comme pour SAT) tout en maximisant la somme des poids des clauses souples satisfaites. Une formulation duale, appelée *Weighted Partial MinSAT* (WP-MinSAT) aura pour objectif de minimiser la somme des poids des clauses souples non-satisfaites. Notre objectif fut alors de formaliser MSCP en problèmes WP-MaxSAT et WP-MinSAT afin d'utiliser les puissants solveurs de la littérature [90, 91] dédiés à ces problèmes qui font l'objet de compétitions mondiales depuis des décennies et affichent donc de très bonnes performances.

### Encodage de MSCP en WP-MaxSAT

**Le problème MaxSAT** L'objectif du problème SAT est de déterminer une interprétation satisfaisant l'ensemble des clauses d'une formule CNF. Mais, lorsqu'une formule n'admet pas de

solution il peut être intéressant de rechercher une interprétation maximisant le nombre de clauses satisfaites. Ce problème, appelé problème MaxSAT, est un problème d'optimisation NP-difficile Papadimitriou (2003).

**Les problèmes Partial MaxSAT et Weighted Partial MaxSAT** Le problème Partial MaxSAT est une extension du problème MaxSAT. Un problème *Partial MaxSAT* considère une CNF dont les clauses sont réparties en deux sous-ensembles : les clauses dures (hards) et les clauses souples (softs). L'objectif du problème Partial MaxSAT est de déterminer une interprétation satisfaisant l'ensemble des clauses dures et maximisant le nombre de clauses souples satisfaites. Lorsque des poids sont associés aux clauses souples et que l'objectif est de maximiser la somme des poids des clauses souples satisfaites, on parle alors du problème *Weighted Partial MaxSAT* (WP-MaxSAT).

**Encodage WP-MaxSAT** La modélisation du problème MSCP vers un formalisme SAT, nécessite de prendre en considération le poids associé à chaque couleur. Nous utiliserons donc le problème WP-MaxSAT pour formuler MSCP. Les clauses dures sont identiques aux clauses de type A, B et C décrites dans le paragraphe 2.2.4 ci-dessus. Les clauses souples sont des clauses unitaires (formées d'un seul littéral). Lors des travaux de thèse de Clément Lecat, nous avons proposé deux encodages différents pour les clauses souples permettant de résoudre MSCP :

Encodage 1 : pour chaque sommet  $i$  et pour chaque couleur  $j$  il existe une clause unitaire  $x_{ij}$  de poids associé  $\Delta(G) + 2 - j$ .

Encodage 2 : pour chaque sommet  $i$  et pour chaque couleur  $j$  il existe une clause unitaire  $\neg x_{ij}$  de poids associé  $j$

Dans les deux cas, maximiser la somme des poids des clauses souples satisfaites, revient à minimiser la somme des couleurs affectées aux sommets du graphe.  $\Delta(G)$  représente le degré du graphe. Le tableau de la figure 2.11 liste l'ensemble des clauses souples suivant les deux encodages, pour le problème WP-MaxSAT sur le graphe de la figure 2.9.

Clauses souples WP-MaxSAT			
Encodage 1		Encodage 2	
clause	poids	clause	poids
$x_{11}$	7	$\neg x_{11}$	1
$x_{12}$	6	$\neg x_{12}$	2
$x_{13}$	5	$\neg x_{13}$	3
$x_{21}$	7	$\neg x_{21}$	1
$x_{22}$	6	$\neg x_{22}$	2
$x_{23}$	5	$\neg x_{23}$	3
$x_{31}$	7	$\neg x_{31}$	1
$x_{32}$	6	$\neg x_{32}$	2
$x_{33}$	5	$\neg x_{33}$	3
$x_{41}$	7	$\neg x_{41}$	1
$x_{42}$	6	$\neg x_{42}$	2
$x_{43}$	5	$\neg x_{43}$	3

FIGURE 2.11 – Ensemble des clauses souples pour MSCP

### Encodage de MSCP en WP-MinSAT

**Le problème MinSAT** Le problème MinSAT est le problème dual de MaxSAT. L'objectif du problème MinSAT est de déterminer une interprétation d'une formule CNF minimisant le nombre de clauses satisfaites. En d'autres termes, maximiser le nombre de clauses insatisfaites. Il est en effet parfois plus facile de déterminer l'insatisfaction d'une instance plutôt que sa satisfaction. Le problème MinSAT a été prouvé NP-difficile dans Kohli & al [92].

**Les problèmes Partial MinSAT et Weighted Partial MinSAT** De la même manière que P-MaxSAT et WP-MaxSAT sont des extensions du problème MaxSAT, Partial MinSAT (P-MinSAT) et Weighted Partial MinSAT (PW-MinSAT) sont tous deux des extensions de MinSAT. Il s'agit de déterminer pour P-MinSAT (resp. WP-MinSAT) une interprétation satisfaisant les clauses dures et minimisant le nombre (resp. la somme des poids) des clauses souples satisfaites.

**Encodage WP-MinSAT** Le problème WP-MinSAT étant le problème dual du problème WP-MaxSAT, l'encodage en est donc très proche. Les clauses dures sont évidemment identiques à la formulation de GCP pour SAT, et les clauses souples sont des clauses unitaires, également proposées suivant deux encodages différents :

Encodage 1 : pour chaque sommet  $i$  et pour chaque couleur  $j$  il existe une clause unitaire  $\neg x_{ij}$  de poids associé  $\Delta(G) + 2 - j$ .

Encodage 2 : pour chaque sommet  $i$  et pour chaque couleur  $j$  il existe une clause unitaire  $x_{ij}$  de poids associé  $j$

Dans les deux cas, minimiser la somme des poids des clauses souples satisfaites, revient à minimiser la somme des couleurs affectées aux sommets du graphe.

Les encodages énoncés ci-dessus pour modéliser MSCP ne sont pas optimaux en taille. La complexité spatiale de la formule CNF générée est en  $O((\Delta(G) + 1) \times 2 |V| + |E|)$ , car le nombre de couleurs envisagé est  $\Delta(G) + 1$ . Cependant, comme proposé dans la thèse de Clément Lecat, il est inutile pour un sommet  $i$  du graphe de considérer plus de  $\min\{s(G), d(i) + 1\}$  couleurs où  $d(i)$  représente le degré du sommet  $i$  et  $s(G)$  la force du graphe. Cette réduction est apportée à l'ensemble des benchmarks sur lesquels nous avons testés les différents encodages. Si  $s(G)$  n'est pas connue, une borne supérieure est utilisée.

### 2.2.5 Synthèse des résultats WP-MaxSAT et WP-MinSAT

Nous avons comparé expérimentalement le comportement des différents encodages sur le problème Weighted Partial MaxSAT et le problème Weighted Partial MinSAT. Pour la résolution du problème WP-MaxSAT, nous avons utilisé la version de 2013 du solveur ISAC de Ansótegui et al. [90] et pour la résolution du problème WP-MinSAT, nous avons considéré la version de 2013 du solveur MinSatz de Li & al. [91].

Cette expérimentation mit en évidence l'efficacité du solveur ISAC pour la résolution du problème WP-MaxSAT. Sur les 76 instances de DIMACS et COLOR02 testées, 34 ont été résolues par ISAC contre 3 pour le solveur MinSatz. L'encodage a également un impact sur les performances des différents solveurs. A titre d'exemple, pour WP-MinSAT, l'instance myciel4 est résolue en moins d'une seconde pour l'encodage 1 alors qu'il aura fallu 7684 secondes pour l'encodage 2. Pour le problème WP-MaxSAT, la tendance est inversée, l'encodage 2 est plus performant que l'encodage 1 notamment pour les graphes ayant une densité supérieure à 0.1. Dans le cas où la densité est faible, aucune tendance ne se dégage.

### 2.2.6 Représentation de l'espace des solutions par Motifs

Que ce soit pour une modélisation en SAT, ou pour le développement d'un algorithme de type branch-and-bound, la résolution optimale de la somme chromatique d'un graphe, nécessite l'utilisation d'un nombre de couleurs à considérer pour définir l'espace des solutions à explorer. Dans l'idéal nous pourrions utiliser la force  $s(G)$  du graphe, mais déterminer cette valeur est un problème NP-difficile. Il est donc primordial de déterminer une borne supérieure de  $s(G)$  de bonne qualité. Or, seules quelques bornes relatives au degré général du graphe étaient connues au début de nos travaux.

Grâce à notre expérience de la décomposition arborescente de graphe, et à la modélisation des solutions partielles sous forme de partitions des sommets du graphe, nous avons proposé une toute nouvelle vision du problème MSCP. En effet, ce qui importe avant tout pour le problème de la somme coloration est la cardinalité des classes couleurs représentées par les blocs des partitions. Or, ces cardinalités sont des entiers. Une partition est donc représentable par une séquence d'entiers.

Nous avons donc défini la notion de *motifs*, qui sont des séquences ordonnées d'entiers, permettant de symboliser l'espace des solutions de manière agrégée. La relation d'ordre définie sur l'ensemble de ces motifs nous a permis de mettre en avant une nouvelle borne inférieure de la somme chromatique  $\Sigma(G)$  et deux nouvelles bornes supérieures de la force  $s(G)$ . Cette dernière a surpassé les résultats de la littérature.

**Définitions des motifs**

Lors de la recherche d'une solution optimale pour MSCP, seule la considération des colorations majeures est nécessaire comme nous l'avons précisé dans la section 2.2.2. Une coloration majeure étant une coloration dont les classes couleurs sont ordonnées par ordre décroissant de leur cardinalité, on associe à chacune de ces colorations une séquence d'entiers eux mêmes décroissants. Une telle séquence est appelée motif et se définit formellement comme suit :

**Définition 26 (Motif)** Soit  $G = (V, E)$  un graphe et  $X = \{X_1, X_2, \dots, X_k\}$  une coloration majeure valide de  $G$ . Un motif est une séquence décroissante d'entiers, notée  $p = \{p[1], p[2], \dots, p[k]\}$ , où chacun de ces entiers est égal à la cardinalité de la classe couleur associée :  $p[i] = |X_i|$  et  $\forall i \in [1 \dots k], p[i] \neq 0$ . La cardinalité du motif  $p$  correspond au nombre de couleurs utilisées par  $X$  :  $|p| = k$ .

La somme coloration associée au motif  $p$  est calculée suivant l'équation 2.3.

$$\sum(p) = 1 \times p[1] + 2 \times p[2] + \dots + k \times p[k] \tag{2.3}$$

L'exemple de la figure 2.12 montre comment trois colorations différentes d'un même graphe, se ramènent à un même motif **(2,1,1)**, dont la somme coloration associée est la somme coloration de la coloration majeure des trois colorations et vaut  $1 \times 2 + 2 \times 1 + 3 \times 1 = 7$ . Dans cet exemple, si "coloration n°2" est symétrique à "coloration n°3", tel n'est pas le cas pour "coloration n°1". Ce simple exemple montre à quel point la représentation sous forme de motifs agrège l'espace des solutions.

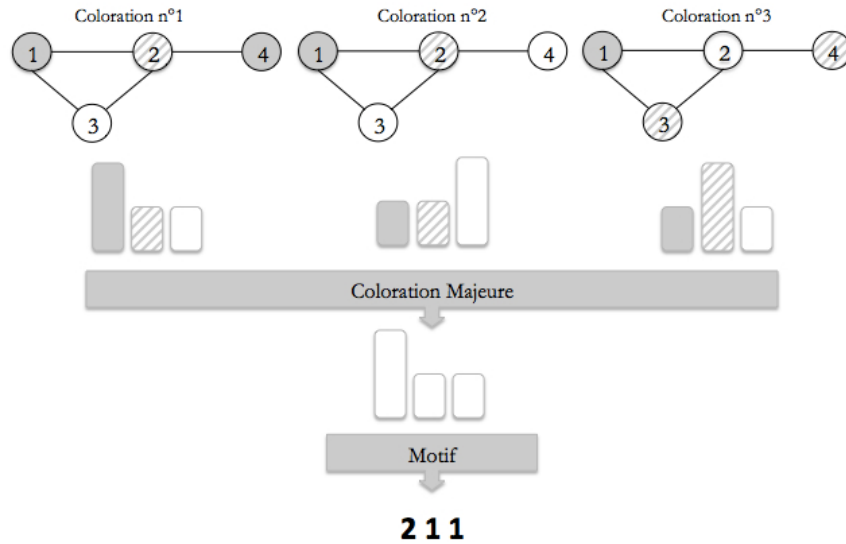


FIGURE 2.12 – Représentation d'un ensemble de colorations par un même motif

Pour un nombre de sommets  $n$  et un nombre de couleurs  $k$  donnés, il est possible de générer l'ensemble des motifs de cardinalité  $k$  que l'on note  $\phi(n, k)$ . L'ensemble  $\phi(n) = \bigcup_{i=1}^{i=k} \phi(n, i)$  est l'ensemble de tous les motifs d'ordre  $n$ . L'ensemble des solutions de tout graphe  $G = (V, E)$  tel que  $|V| = n$  se trouve représenté par un sous ensemble des motifs de  $\phi(n)$ . Ainsi, sur la figure 2.13 les motifs  $(2, 1, 1)$  et  $(1, 1, 1, 1)$  représentent à eux seuls toutes les solutions valides du graphe.



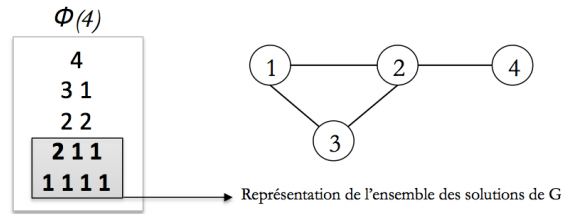


FIGURE 2.13 – Espace des solutions et motifs

Notre approche consiste donc à rechercher où se trouvent les solutions d'un graphe  $G$  parmi les motifs de  $\phi(n)$  afin d'approcher la solution optimale de la force du graphe et de sa somme chromatique. Or, nous remarquons que la construction de l'ensemble  $\phi(n)$  est similaire à la construction de l'ensemble des partitions des sommets du graphe. Ce nombre croît de manière exponentielle en fonction de  $n$ , il n'est donc pas envisageable de générer tous les motifs de  $\phi(n)$ . Nous nous basons donc sur la *relation de dominance* existante entre les motifs de  $\phi(n)$  qui nous permet alors de n'explorer qu'un très petit sous-ensemble de motifs.

La relation de dominance, notée  $\succeq$ , sur l'ensemble des motifs de  $\phi(n)$  a été introduite dans [93, 94]. Nous avons adapté cette relation de dominance à notre approche.

**Définition 27 (Relation de dominance)** Soient  $p$  et  $q$  deux motifs de  $\phi(n)$ . On dit que  $p$  domine  $q$ , noté  $p \succeq q$ , si et seulement si  $\forall t$  tel que  $1 \leq t \leq \min\{|p|, |q|\}$ , la relation suivante est vérifiée :

$$\sum_{i=1}^t p[i] \geq \sum_{i=1}^t q[i].$$

À titre d'exemple, soient  $p$  et  $q$  deux motifs, tels que  $p = (5, 2, 1)$  et  $q = (4, 3, 1)$ .

- Pour  $t = 1$  :  $5 \geq 4$ ;
- Pour  $t = 2$  :  $5 + 2 \geq 4 + 3$ ;
- Pour  $t = 3$  :  $5 + 2 + 1 \geq 4 + 3 + 1$ .

On peut donc en conclure que  $p \succeq q$ .

Cependant, la relation de dominance entre deux motifs est une relation d'ordre partiel. Supposons maintenant que  $p = (9, 3, 3)$  et  $q = (8, 6, 1)$ . Alors :

- Pour  $t = 1$  :  $9 \geq 8$ ;
- Pour  $t = 2$  :  $9 + 3 < 8 + 6$ ;

On dit dans ce cas que  $p$  et  $q$  sont incomparables. L'intérêt de la relation de dominance entre les motifs est qu'elle permet de hiérarchiser les motifs entre eux relativement à leur somme coloration associée, comme le montre la propriété 3

**Propriété 3** Soient  $p$  et  $q$  deux motifs, tels que  $p \succeq q$  alors  $\sum(p) \leq \sum(q)$ .

Cette propriété a été démontrée dans la thèse de Clément Lecat et généralisée au problème OCCP. Afin d'exploiter cette relation de dominance entre les motifs, nous avons défini une relation d'ordre total sur l'ensemble  $\phi(n)$ . Pour cela, nous considérons que les ensembles  $\phi(n, k) \subset \phi(n)$  sont triés par ordre croissant de  $k$  et que les motifs d'un sous-ensemble  $\phi(n, k)$  sont triés par ordre lexicographique décroissant.

**Définition 28 (Ordre total sur  $\phi(n)$ )** Nous notons  $p_k^i$  le  $i$ -ème motif dans l'ordre lexicographique décroissant de  $\phi(n, k)$ . Soit  $p_k^i$  et  $p_{k'}^j$  deux motifs de  $\phi(n)$ . Le motif  $p_k^i$  précède  $p_{k'}^j$  dans  $\phi(n)$ , si  $k > k'$  ou si  $k = k'$  et  $i < j$ .

Grace à cet ordre et la relation de dominance nous avons construit deux bornes supérieures pour la force du graphe, une borne algébrique  $UB_A$  et une borne algorithmique  $UB_S$ .

### Bornes Supérieures de la force du graphe

Le principe de construction des bornes  $UB_A$  et de  $UB_S$  est le suivant : étant donné un motif  $p$  associé à une coloration valide  $X$  d'un graphe  $G = (V, E)$ , avec  $n = |V|$ , déterminer le nombre de couleurs  $k_{max}$ , tel que  $k_{max} \in \{1, \dots, n\}$ , pour lequel tout motif  $q$  de cardinalité supérieure à  $k_{max}$  est dominé par  $p$ , ainsi que le spécifie l'équation 2.4.

$$\forall q \in \Omega_{k_{max}} = \bigcup_{x=k_{max}}^n \phi(n, x), \quad p \succeq q \quad (2.4)$$

**Borne supérieure  $UB_A$  :** La borne algébrique  $UB_A$  s'appuie sur une propriété que nous avons démontrée dans [8] et qui stipule que  $\forall k \in \{1, \dots, n\}$ ,  $p_k^1$  domine tous les motifs  $q \in \Omega_k$ . En conséquence, si on considère une coloration valide  $X$  de  $G = (V, E)$  et  $p_k^1$  le motif de  $\phi(n)$  tel que  $k$  est le plus petit nombre de couleurs pour lequel  $\sum(p_k^1) \geq \sum(X)$  qui s'écrit aussi  $\sum(p_k^1) - \sum(X) \geq 0$ . Alors  $k$  est une borne supérieure pour la force. Comment déterminer  $k$ ? Structurellement comme  $p_k^1$  est constitué d'une classe couleur de  $(n - k + 1)$  sommets et de  $(k - 1)$  classes couleurs de 1 sommet, il est facile de calculer  $\sum(p_k^1)$  suivant l'équation 2.5. Or, la proposition  $\sum(p_k^1) - \sum(X) \geq 0$  est vraie si  $k$  est supérieur à la seconde racine du polynôme, c'est à dire si  $k$  vérifie l'équation 2.6. Ce qui rend la force  $s(G)$  supérieurement bornée par  $UB_A$  dont l'expression est donnée par l'équation 2.7.

$$\sum(p_k^1) = (n - k + 1) + \sum_{x=2}^k x = \frac{1}{2}k^2 - \frac{1}{2}k + n \quad (2.5)$$

$$k \geq \frac{1 + \sqrt{1 + 8(\sum(X) - n)}}{2} \quad (2.6)$$

$$UB_A = \max\{\lceil \frac{1 + \sqrt{1 + 8(\sum(X) - n)}}{2} \rceil - 1, |X|\} \quad (2.7)$$

**Borne supérieure  $UB_S$  :** Il est indubitable que la coloration du graphe  $G$  représentée par  $p_k^1$  est dans le cas général loin d'une coloration valide de  $G$ . Pour cette raison, et afin d'améliorer la qualité de la borne supérieure de  $s(G)$ , nous proposons de considérer la cardinalité du stable maximum lors de l'exploration de  $\phi(n)$ . En effet, aucune coloration valide de  $G$  ne peut avoir de classe couleur de cardinalité supérieure à celle du stable maximum noté  $\alpha(G)$ . Tous les motifs  $p$  pour lesquels  $p[1] \geq \alpha(G)$  peuvent donc être exclus de l'espace de recherche.

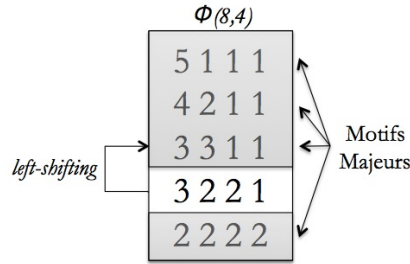
Maintenant considérons l'opération appelée  $q = \text{left-shifting}(i, j, p)$  qui consiste à partir d'un motif  $p \in \phi(n, k)$  à le transformer en un motif  $q \in \phi(n, k)$  comme suit :

- soit  $1 \leq i \leq j \leq k$  alors
  - $\forall x \in [1 \dots k], x \neq i, x \neq j, q[x] = p[x]$
  - $q[i] = p[i] + 1$
  - $q[j] = p[j] - 1$

Cette opération, qui revient à transférer un sommet de la classe couleur  $X_j$  vers la classe couleur  $X_i$ , implique que  $\sum(q) \leq \sum(p)$ . Mais il résulte de la propriété des motifs à représenter des colorations majeures, que le  $\text{left-shifting}()$  n'est pas toujours applicable : si la transformation entraîne  $q[i] < q[i - 1]$  ou si  $q[i] = 0$ , alors  $q$  n'est pas un motif. Nous nous appuyons sur cette constatation pour introduire la notion de *motif majeur* dans la définition 29.

**Définition 29 (Motif majeur)** Soit  $p \in \phi(n, k)$ . Si  $\nexists q \in \phi(n, k)$  tel que  $q = \text{left-shifting}(i, j, p)$ , et  $2 \leq i \leq j \leq k$  alors  $p$  est appelé motif majeur.

Considérons  $\phi(8, 4)$ , l'ensemble des motifs possibles pour un graphe de 8 sommets, et pour 4 couleurs comme illustré par la figure 2.14. Tous les motifs de  $\phi(8, 4)$  sont majeurs, à l'exception de  $(3, 2, 2, 1)$ , car  $(3, 3, 1, 1) = \text{left-shifting}(2, 3, (3, 2, 2, 1))$ .

FIGURE 2.14 – Motifs majeurs dans  $\phi(8, 4)$ .

A partir de la notion de motif majeur, nous avons énoncé et prouvé un ensemble de propriétés, sur lesquelles nous nous sommes appuyés pour construire un algorithme polynomial, qui calcule une borne supérieure  $UB_S$  de la force d'un graphe quelconque [8]. La principale propriété (propriété 4) établit une relation de dominance entre un motif majeur  $p_k^i$  et les motifs lui succédant suivant l'ordre total défini sur  $\phi(n)$ .

**Propriété 4** Soient  $k$  et  $k'$  deux entiers tels que  $k < k'$  et  $\Omega_{k'} = \bigcup_{x=k'}^n \phi(n, x)$ . Si  $p_k^i$  est un motif majeur, alors  $\forall q \in \Omega_{k'}$  tel que  $q[1] \leq p_k^i[1]$ ,  $p_k^i \succeq q$ .

A partir de la propriété 4, nous pouvons établir qu'étant donnée une coloration  $X$  d'un graphe  $G$ , si nous déterminons le plus petit nombre de couleurs  $k$ , pour lequel il existe un motif majeur  $p_k^i$  tel que  $p_k^i[1] = \alpha(G)$  et tel que  $\sum(p_k^i) \geq \sum(X)$ , alors il n'existe pas de meilleure coloration pour MSCP, utilisant plus de  $k$  couleurs. Par conséquent, la force du graphe  $G$  est donc bornée supérieurement par  $UB_S = k - 1$ .

L'algorithme  $UB_S$  à partir d'une coloration  $X$  de  $G$ , de la cardinalité du stable maximum  $\alpha(G)$  ou d'une borne supérieure de celle-ci, calcule la borne supérieure de  $s(G)$  en  $O(n^2)$ . A partir de  $k = |X|$ , l'algorithme recherche de manière incrémentale un motif majeur  $p_k^i$ , comme ci-dessus mentionné, jusqu'à ce que  $\sum(p_k^i) \geq \sum(X)$ . Il retourne alors  $k - 1$ .

### 2.2.7 Synthèse des résultats Motifs et bornes supérieures de la force

Nous avons comparé  $UB_A$  et  $UB_S$  aux bornes proposées dans [95]. Peu de travaux existaient sur ce problème. Pour déterminer  $\alpha(G)$ , nous avons utilisé le solveur IncMaxCLQ proposé dans Li et al. [96]. Bien que le problème de stable maximum soit NP-difficile, ce solveur nous a permis dans 90% des instances d'obtenir  $\alpha(G)$  en moins de dix minutes.

Ces expérimentations ont montré que  $UB_A$ , malgré qu'aucune information structurelle ne soit considérée hormis une borne supérieure de la somme chromatique, permet d'améliorer la borne supérieure de la force pour 43 instances sur 74 (58%).  $UB_S$ , améliore considérablement la borne supérieure de la force pour 70 instances sur 74, soit 94% des instances testées. Le gain est parfois très élevé. A titre d'exemple, pour les graphes de la famille inithx ( $x=1,2,3$ ), les résultats de la littérature sont respectivement 278, 286 et 287, pour  $UB_A$  ils sont de 75, 54 et 53 et pour  $UB_S$  72, 48 et 48. Ces gains permettent de réduire le facteur exponentiel (dans ce cas de 200 couleurs) de la taille de l'espace de recherche pour la résolution de MSCP.

Pour 8 instances (le450-5c, le450-5d, myciel3, queen5-5, queen7-7, queen8-12, flat300-20-0 et flat1000-50-0) nous avons la relation  $UB_S = \chi(G)$  ce qui signifie que l'optimalité est atteinte :  $UB_S = s(G)$ .

Les 6 instances pour lesquelles  $UB_S$  n'améliore pas les scores, sont celles dont le stable maximum n'a pu être déterminé par IncMaxCLQ.

### Borne Inférieure de la somme chromatique

Les différentes propriétés élaborées autour des motifs ont également débouché sur l'élaboration d'une borne inférieure algébrique, appelée  $LBM\Sigma$ , pour la somme chromatique du graphe.  $LBM\Sigma$

est le fruit de l'analyse de la borne proposée par [86], et des propriétés énoncées sur les motifs majeurs. Cette borne nommée  $LB_{kok}$ , comme le montre l'équation 2.8 fait intervenir  $n$  le nombre de sommets du graphe et  $\chi(G)$  son nombre chromatique.

$$LB_{kok} = n - \chi(G) + \frac{\chi(G) \times (\chi(G) + 1)}{2} \quad (2.8)$$

D'après la propriété 4, nous savons que le motif majeur  $p_k^1$  domine tous les motifs qui lui succèdent dans  $\phi(n)$ . Par là même, comme  $\chi(G)$  est le plus petit nombre de couleurs nécessaire à la coloration de  $G$ , et que le motif  $p_{\chi(G)}^1$  domine tous ses successeurs, alors  $\sum(p_{\chi(G)}^1)$  est une borne inférieure de la somme chromatique. Or,  $p_{\chi(G)}^1$  correspond au motif  $(a_1, a_2, \dots, a_{\chi(G)})$  avec  $a_1 = n - \chi(G)$  et  $\forall i \geq 2, a_i = 1$ . Suivant cette structure, on a  $\sum(p_{\chi(G)}^1) = LB_{kok}$ . Mais comme énoncé dans le paragraphe précédent, tous les motifs dont le premier entier est supérieur à  $\alpha(G)$  ne sont pas à considérer dans l'espace des solutions pour  $G$ . De ce fait le premier motif dans  $\phi(n)$  qui pourrait correspondre à une solution pour  $G$  est le premier motif majeur  $p_{\chi(G)}^i$  pour lequel  $p_{\chi(G)}^i[1] = \alpha(G)$ . La somme coloration associée,  $\sum(p_{\chi(G)}^i)$  est une borne inférieure pour la somme chromatique du graphe. Ce motif majeur  $p_{\chi(G)}^i$  correspond au motif  $(a_1, a_2, \dots, a_{\chi(G)})$  tel que  $a_i = \alpha(G), \forall i = 1, \dots, \beta, a_{\beta+1} = n - \beta \times \alpha(G) - \chi(G) + \beta - 1$  et  $a_i = 1, \forall i = \beta + 2, \dots, \chi(G)$ . De part cette structure, illustrée par la figure 2.15, la borne  $LBM\Sigma$  est directement calculée suivant l'équation 2.9.

$$\begin{aligned} \text{Soit } G=(V,E) \text{ tel que } |V|=n, \quad \chi(G) = \chi, \quad \alpha(G) = \alpha \text{ et } \beta = \lfloor \frac{n-\chi}{\alpha-1} \rfloor \\ LBM\Sigma = \alpha \times \frac{\beta \times (\beta + 1)}{2} \\ + n - \beta \times \alpha - (\chi - \beta - 1) \times (\beta + 1) \\ + \sum_{j=\beta+2}^{\chi} j \end{aligned} \quad (2.9)$$

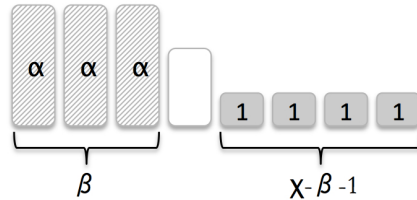


FIGURE 2.15 – Structure d'un motif majeur  $p$  avec  $p[1] = \alpha(G)$

### 2.2.8 Synthèse des résultats Motifs et borne inférieure de la somme chromatique :

$LBM\Sigma$ , fut comparée aux meilleurs résultats connus pour la borne inférieure de la somme coloration de la littérature. Ces bornes sont basées sur une heuristique décomposant le graphe en une partition de cliques [5, 75], comme mentionné dans le paragraphe sur l'algorithme MA-MSCP. Les résultats obtenus montraient que  $LBM\Sigma$  a permis d'améliorer strictement les résultats de la littérature pour 33 instances sur 74. Pour 10 autres instances, la meilleure borne connue fut atteinte. De plus grâce à  $LBM\Sigma$  l'optimalité de la somme chromatique pour quatre instances (flat1000-50-0, flat300-20-0, le450-5c et le450-5d) a pu être démontrée. Cependant, pour 31 instances notre approche est moins efficace. L'une des raisons est que dans certains cas, une méthode de partitionnement en cliques peut donner une borne inférieure extrêmement proche de l'optimum. Il est donc très difficile d'améliorer cette dernière. De plus, dans le cas des graphes faiblement denses, la partition en cliques du graphe permet une meilleure considération des propriétés structurelles.

## 2.3 Conclusion

Le problème de coloration de graphe et plus particulièrement celui du nombre chromatique est un problème auquel il fut difficile de se confronter, tant la littérature est riche de résultats, que ce soit en théorie des graphes ou en algorithmique des graphes. Ce problème a durant toutes ces décennies suscité un vif intérêt de la part des chercheurs, car je pense qu'au delà du nombre important de ses applications, il constitue un matériau de réflexion et d'expérimentation ludique et mobilisateur. Ce problème est en effet très simple à comprendre, nous en avons même édité une version papier sous forme de jeu pour les enfants lors de la fête de la science, mais tellement difficile à résoudre.

Mon intérêt pour le problème de coloration est une suite logique de mon travail sur le problème de fiabilité des réseaux résolu par une méthode exacte de décomposition arborescente du graphe. Ces deux problèmes partageant en effet la même modélisation de leurs solutions partielles.

J'ai eu la chance de co-encadrer trois thèses autour de ces problèmes. Premièrement la thèse de Florence Mendes de 2001 à 2005, en collaboration avec le Pr. Aziz Moukrim (HeuDiaSyC/UTC), lors de laquelle nous avons développé une décomposition linéaire pour le problème du nombre chromatique, ainsi que des opérateurs de pré-traitement et des améliorations de la méthode de base. Ensuite, nous avons travaillé autour de la thèse de Kaoutar Sghiouer de 2007 à 2011, en collaboration avec le Pr. Aziz Moukrim et Yu Li (MIS/UPJV) sur le problème de la somme chromatique, pour lequel nous avons proposé des méthodes gloutonnes et une métaheuristique de type évolutionnaire. Enfin, les derniers travaux furent l'objet d'une collaboration avec le Pr. Chumin Li (MIS/UPJV) à travers le co-encadrement de la thèse de Clément Lecat de 2014 à 2017. Notre étude s'est alors portée sur la résolution du problème de la somme chromatique des graphes, par une approche exacte. Nous avons proposé des algorithmes de type branch-and-bound basés sur une représentation des sous-solutions sous forme de partitions de sommets, ainsi qu'une formulation MaxSAT et MinSAT du problème. Nous avons également abordé ce problème avec un nouvel angle en proposant une modélisation de l'ensemble des solutions sous forme de "motifs" sur lesquels nous avons énoncé quelques propriétés. Cette dernière approche nous a valu d'obtenir d'excellents résultats sur la borne inférieure de la somme et sur la borne supérieure de la force d'un graphe. Nos recherches se poursuivent sur ce dernier point, car beaucoup de pistes restent à explorer avec cette nouvelle vision de l'ensemble des solutions permettant de le réduire et de le hiérarchiser.

L'ensemble de ces travaux ont été publiés dans des revues internationales et des conférences internationales de haut rang. Une sélection de quatre articles ci-dessous mentionnés se trouvent dans les annexes de ce mémoire.

- [4] Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. *Computers & OR*, 33 :2189-2207, 2006.
- [5] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, and Yu Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663-670, 2010.
- [7] Clément Lecat, Corinne Lucet, and Chu-Min Li. New lower bound for the minimum sum coloring problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4-9, 2017, San Francisco, California, USA., pages 853-859, 2017
- [8] Clément Lecat, Corinne Lucet, and Chu-Min Li. Minimum sum coloring problem : Upper bounds for the chromatic strength. *Discrete Applied Mathematics*, 233 :71-82, 2017.

# Conclusion générale et perspectives

Ce document est une présentation synthétique et non exhaustive de mes activités de recherche sur presque trente ans. Il fait état des problèmes combinatoires que j'ai pu étudier, comme le problème de fiabilité des réseaux, le problème du nombre chromatique et le problème de la somme chromatique.

Le problème de la fiabilité fut l'objet de collaborations avec les professeurs Jacques Carlier (HeuDiaSyC/UTC) et Nikolaos Limnios (LMAC/UTC). Nous avons proposé une modélisation et des algorithmes de résolution de ce problème, basés premièrement sur une décomposition arborescente du graphe, puis sur les puissantes structures de représentation que sont les diagrammes de décision binaire (BDD). Nous avons montré que des réseaux de faible largeur linéaire pouvaient être traités très efficacement par ce type de méthode, dont la complexité est exponentielle, non pas relativement à la taille du graphe, mais relativement à sa largeur linéaire [1, 2]. Nous avons également fait la relation entre la complexité d'un BDD associé à un réseau et la largeur linéaire de celui-ci [3, 12, 45]. Ces BDD nous ont permis d'obtenir des résultats très performants comparés à ceux de la littérature pour un problème d'optimisation de conception de réseau avec contrainte de fiabilité [47]. Nous ne travaillons plus actuellement sur ce problème, mais nos travaux sont toujours largement cités à ce jour, car ils constituent une base de travail pour de nombreux chercheurs.

Ces résultats nous ont amené à examiner quels autres problèmes combinatoires pourraient être traités de manière efficace par une décomposition linéaire. Notre choix se porta sur un problème central en optimisation combinatoire, celui du nombre chromatique d'un graphe. Notre motivation était liée à la structure du problème, qui se modélise de manière identique au problème de fiabilité des réseaux. Ces travaux furent menés en collaboration avec les professeurs Jacques Carlier et Aziz Moukrim (HeuDiaSyC/UTC). Nous avons proposé une amélioration de la décomposition linéaire par la combinaison d'opérateurs de pré-traitement du graphe et une technique de branch-and-bound. Cette méthode fut comparée aux résultats de la littérature, et en accord avec sa complexité théorique, celle-ci s'est montrée efficace sur les graphes de faible largeur linéaire, ou sur les graphes de forte densité [49]. Les opérateurs de pré-traitement ont donné de très bons résultats, allant pour certaines familles de graphes jusqu'à les réduire totalement [50].

Nous nous sommes ensuite intéressés à un autre problème de coloration, celui de la somme chromatique d'un graphe, plus récemment apparu. Ces travaux se divisent en deux parties. La première concerne une collaboration avec Aziz Moukrim et Yu Li (MIS/UPJV) durant laquelle nous avons abordé le problème par l'élaboration de méthodes approchées comme des algorithmes gloutons (MDSAT et MRLF [69]) et une métaheuristique de type évolutionnaire (MA-MSCP [71]). Ces méthodes ont également fourni des bornes inférieures de la somme chromatique, à partir desquelles nous avons défini un nouveau problème d'optimisation de partitionnement du graphe en cliques (PCMSCP [5]).

La deuxième partie des travaux consacrés au problème du nombre chromatique, fut dédiée à la résolution par des méthodes exactes, en collaboration avec le Professeur Chumin Li (MIS/UPJV). Tout d'abord en proposant une modélisation de celui-ci sous forme de problème SAT, adapté aux

problèmes d'optimisation, à savoir deux encodages pour chacun des problèmes Weighted Partial MaxSAT et Weighted Partial MinSAT. Le but étant d'utiliser les solveurs associés de la littérature pour ces problèmes. Puis, nous avons construit deux algorithmes de type branch-and-bound [72]. Le premier basé sur un branchement classique sur les couleurs et le second sur un branchement basé sur les ensembles frontières. Les résultats ne furent pas très concluants. En effet, la force du graphe pouvant être arbitrairement grande, il est nécessaire en l'absence de borne supérieure de qualité de considérer un trop grand nombre de couleurs dans l'exploration de l'ensemble des solutions. Cet obstacle nous a conduit à analyser cet ensemble de solutions, pour lequel nous avons proposé une représentation agrégée sous forme de motifs. Grâce à cette représentation nous avons obtenus de très bons résultats, pour la borne supérieure de la force [7], ainsi que pour la borne inférieure de la somme chromatique [8]. Les perspectives d'une telle approche sont nombreuses. L'emploi des motifs a pour principe d'utiliser les propriétés structurelles du graphe pour restreindre l'espace de recherche. Or, seul le stable maximum du graphe a été exploité pour le moment. Nous souhaitons introduire d'autres mesures liées à la nature du graphe pour améliorer encore ces résultats. Mais aussi aborder d'autres problèmes combinatoires avec ce nouvel angle. Les algorithmes de branch-and-bound peuvent également être perfectionnés en améliorant les bornes de coupe. La collaboration entamée il y a cinq ans avec le professeur Chumin Li se poursuivra donc ces prochaines années sur ce thème prometteur.

Une autre partie de mes recherches n'a pas été développée dans ce document. Il s'agit des problèmes de logistique et plus spécifiquement des problèmes de planification sur lesquels nous travaillons en collaboration avec Laure Brisoux-Devendeville (MIS/UPJV). Cette activité prend actuellement de l'ampleur avec deux projets au sein desquels nous développons des algorithmes pour la planification de sessions de formation pour le centre de pédagogie active SimUSanté (CHU-Amiens), et pour l'optimisation du parcours patient en milieu hospitalier en contrat avec l'entreprise Evolucare. Deux projets avec une problématique commune, mais des contraintes et des objectifs différents qui s'inscrivent parfaitement dans la coloration e-santé du laboratoire MIS. Nous avons élaboré des modèles mathématiques et nous sommes en phase d'évaluation des premières méthodes constructives [97].

# Bibliographie

- [1] Jacques Carlier and Corinne Lucet. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics*, 65(1) :141 – 156, 1996. First International Colloquium on Graphs and Optimization.
- [2] Corinne Lucet, Jean-Francois Manouvrier, and Jacques Carlier. Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs. *Algorithmica*, 27(3) :316–336, 2000.
- [3] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Trans. Reliability*, 56(3) :506–515, 2007.
- [4] Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. *Computers & operations research*, 33(8) :2189–2207, 2006.
- [5] Aziz Moukrim, K. Sghiouer, Corinne Lucet, and Y. Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663–670, 2010.
- [6] Anas Abdoul Soukour, Laure Devendeville, Corinne Lucet, and Aziz Moukrim. A memetic algorithm for staff scheduling problem in airport security service. *Expert Syst. Appl.*, 40(18) :7504–7512, 2013.
- [7] Clément Lecat, Corinne Lucet, and Chu-Min Li. Minimum sum coloring problem : Upper bounds for the chromatic strength. *Discrete Applied Mathematics*, 233 :71–82, 2017.
- [8] Clément Lecat, Corinne Lucet, and Chu-Min Li. New lower bound for the minimum sum coloring problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 853–859, 2017.
- [9] Michael O. Ball. Complexity of network reliability computations. *Networks*, 10(2) :153–165, 1980.
- [10] J. Scott Provan and Michael O. Ball. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3) :516–526, 1984.
- [11] Tadashi Dohi, Shunji Osaki, and Katsushige Sawaki. *Recent Advances in Stochastic Operations Research*. WORLD SCIENTIFIC, 2007.
- [12] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. Probability of connection in regular stochastic networks. *SIGMETRICS Performance Evaluation Review*, 34(2) :9–10, 2006.
- [13] S. Hasanuddin Ahmad. A simple technique for computing network reliability. *IEEE Transactions on Reliability*, R-31(1) :41 – 44, 1982.
- [14] S Hasanuddin Ahmad. Simple enumeration of minimal cutsets of acyclic directed graph. 37 :484 – 487, 01 1989.
- [15] M. O. Locks. A minimizing algorithm for sum of disjoint products. *IEEE Transactions on Reliability*, R-36(4) :445–453, Oct 1987.
- [16] Salim Hariri and C. S. Raghavendra. Syrel : A symbolic reliability algorithm based on path and cutset methods. *IEEE Transactions on Computers*, C-36(10) :1224–1232, 1987.
- [17] Appajosyula Satyanarayana and R. Kevin Wood. A linear-time algorithm for computing k-terminal reliability in series-parallel networks. *SIAM J. Comput.*, 14(4) :818–832, 1985.
- [18] R. Kevin Wood. A factoring algorithm using polygon-to-chain reductions for computing k-terminal network reliability. *Networks*, 15(2) :173–190, 1985.



- [19] Appajosyula Satyanarayana, R. Kevin Wood, Leonidas Camarinopoulos, and G. Pampoukis. Note on "a linear-time algorithm for computing k-terminal reliability in a series-parallel network". *SIAM J. Comput.*, 25(2) :290, 1996.
- [20] Francis T. Boesch, Appajosyula Satyanarayana, and Charles L. Suffel. A survey of some network reliability analysis and synthesis results. *Networks*, 54(2) :99–107, 2009.
- [21] A. Rosenthal. Computing the reliability of complex networks. *SIAM Journal on Applied Mathematics*, 32(2) :384–393, 1977.
- [22] Arnon Rosenthal. Series-parallel reduction for difficult measures of network reliability. *Networks*, 11(4) :323–334, 1981.
- [23] Olympia R. Theologou and Jacques Carlier. Factoring and reductions for networks with imperfect vertices. 40 :210 – 217, 07 1991.
- [24] Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory, Ser. B*, 35(1) :39–61, 1983.
- [25] Neil Robertson and Paul D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *J. Algorithms*, 7(3) :309–322, 1986.
- [26] Hans L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36 :116–125, 1988.
- [27] Hans L. Bodlaender. Treewidth : Algorithmic techniques and results. In *MFCS*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1997.
- [28] Hans L. Bodlaender. Treewidth of graphs. In *Encyclopedia of Algorithms*, pages 2255–2257. 2016.
- [29] Bruno Courcelle. The monadic second-order logic of graphs IV : definability properties of equational graphs. *Ann. Pure Appl. Logic*, 49(3) :193–255, 1990.
- [30] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybern.*, 11(1-2) :1–21, 1993.
- [31] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1) :11–24, 1989.
- [32] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2) :277–284, 1987.
- [33] Godfrey H Hardy and Srinivasa Ramanujan. Asymptotic formulæ in combinatory analysis. *Proceedings of the London Mathematical Society*, 2(1) :75–115, 1918.
- [34] Sheldon B. Akers Jr. Binary decision diagrams. *IEEE Trans. Computers*, 27(6) :509–516, 1978.
- [35] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8) :677–691, 1986.
- [36] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3) :293–318, 1992.
- [37] Olivier Coudert and Jean Christophe Madre. Implicit and incremental computation of primes and essential primes of boolean functions. In *Proceedings of the 29th Design Automation Conference, Anaheim, California, USA, June 8-12, 1992.*, pages 36–39, 1992.
- [38] Antoine Rauzy. New algorithms for fault trees analysis. *Reliability Engineering & System Safety*, 40(3) :203 – 211, 1993.
- [39] Nikolaos Limnios. *Arbres de défaillance*. 1991.
- [40] Xinyu Zang, Hairong Sun, and Kishor S. Trivedi. A bdd-based algorithm for reliability evaluation of phased mission system. *IEEE Transactions on Reliability*, 48 :50–60, 1999.
- [41] Kyoko Sekine and Hiroshi Imai. A unified approach via bdd to the network reliability and path numbers, 1995.
- [42] Fu-Min Yeh, Shyue-Kung Lu, and Sy-Yen Kuo. Obdd-based evaluation of k-terminal network reliability. *IEEE Trans. Reliability*, 51(4) :443–451, 2002.
- [43] Sean Reed, Magnus Löfstrand, and John Andrews. An efficient algorithm for computing exact system and survival signatures of k-terminal network reliability. *Reliability Engineering & System Safety*, 185 :429 – 439, 2019.

- [44] Jun Kawahara, Koki Sonoda, Takeru Inoue, and Shoji Kasahara. Efficient construction of binary decision diagrams for network reliability with imperfect vertices. *Reliability Engineering & System Safety*, 188 :142 – 154, 2019.
- [45] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. Computing all-terminal reliability of stochastic networks with binary decision diagrams. In *International Symposium on Applied Stochastic Models and Data Analysis, ASMDA 2005, Brest, France, May, 2005, Proceedings*, pages 1468–1474, 2005.
- [46] Mark John Somers and Dee Birnbaum. Work-related commitment and job performance : it’s also the nature of the performance that counts. *Journal of Organizational Behavior*, 19(6) :621–634, 1998.
- [47] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. A bdd-based heuristic algorithm for design of reliable networks with minimal cost. In *Mobile Ad-hoc and Sensor Networks, Second International Conference, MSN 2006, Hong Kong, China, December 13-15, 2006, Proceedings*, pages 244–255, 2006.
- [48] Fulya Altiparmak, Berna Dengiz, and Alice E. Smith. Optimal design of reliable computer networks : A comparison of metaheuristics. *J. Heuristics*, 9(6) :471–487, 2003.
- [49] Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. *Computers & OR*, 33 :2189–2207, 2006.
- [50] Corinne Lucet, Florence Mendes, and Aziz Moukrim. Pre-processing and linear-decomposition algorithm to solve the k-colorability problem. In *Experimental and Efficient Algorithms, Third International Workshop, WEA 2004, Angra dos Reis, Brazil, May 25-28, 2004, Proceedings*, pages 315–325, 2004.
- [51] Daniel Brélaz. New methods to color vertices of a graph. *Commun. ACM*, 22(4) :251–256, 1979.
- [52] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4) :344–354, 1996.
- [53] Enrico Malaguti, Michele Monaci, and Paolo Toth. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2) :174 – 190, 2011.
- [54] Isabel Méndez-Díaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5) :826 – 847, 2006. IV ALIO/EURO Workshop on Applied Combinatorial Optimization.
- [55] Francine Herrmann and Alain Hertz. Finding the chromatic number by means of critical graphs. *J. Exp. Algorithmics*, 7 :10–, December 2002.
- [56] Allen Van Gelder. Another look at graph coloring via propositional satisfiability. *Discrete Applied Mathematics*, 156(2) :230 – 243, 2008. Computational Methods for Graph Coloring and it’s Generalizations.
- [57] Zhaoyang Zhou, Chu-Min Li, Chong Huang, and Ruchu Xu. An exact algorithm with learning for the graph coloring problem. *Computers & Operations Research*, 51 :282 – 301, 2014.
- [58] F.T Leighton. A graph coloring algorithm for large scheduling problems. *Journal of research of the national institute of standards and technology*, 84(6) :251–256, 1979.
- [59] José Luis González-Velarde and Manuel Laguna. Tabu search with simple ejection chains for coloring graphs. *Annals of Operations Research*, 117(1) :165–174, Nov 2002.
- [60] R. Dorne and J.K. Hao. *Tabu Search for Graph Coloring, T-Colorings and Set T-Colorings*, pages 77–92. Springer US, Boston, MA, 1999.
- [61] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4) :345–351, Dec 1987.
- [62] M. Chams, A. Hertz, and D. de Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2) :260 – 266, 1987. Third EURO Summer Institute Special Issue Decision Making in an Uncertain World.
- [63] Zhipeng Lü and Jin-Kao Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1) :241 – 250, 2010.

- [64] Guangzhao Cui, Limin Qin, Sha Liu, Yanfeng Wang, Xuncaizhang, and Xianghong Cao. Modified pso algorithm for solving planar graph coloring problem. *Progress in Natural Science*, 18(3) :353 – 357, 2008.
- [65] Charles Fleurent and Jacques A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63(3) :437–461, Jun 1996.
- [66] Y. Jin and J. Hao. Solving the latin square completion problem by memetic graph coloring. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2019.
- [67] Raja Marappan and Gopalakrishnan Sethumadhavan. Solution to graph coloring using genetic and tabu search procedures. *Arabian Journal for Science and Engineering*, 43(2) :525–542, Feb 2018.
- [68] Mehdi Rezapoor Mirsaleh and Mohammad Reza Meybodi. A michigan memetic algorithm for solving the vertex coloring problem. *Journal of Computational Science*, 24 :389 – 401, 2018.
- [69] Yu Li, Corinne Lucet, Aziz Moukrim, and Kaoutar Sghiouer. Greedy algorithms for the minimum sum coloring problem. In *Logistique et transports*, pages LT–027, 2009.
- [70] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, and Yu Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663–670, 2010.
- [71] Kaoutar Sghiouer, Li Yu, Corinne Lucet, and Aziz Moukrim. A memetic algorithm for the minimum sum coloring problem. In *Conférence Internationale en Recherche Opérationnelle (CIRO'2010)*, 2010.
- [72] Clément Lecat, Chu-Min Li, Corinne Lucet, and Yu Li. Exact methods for the minimum sum coloring problem. In *Doctoral Program of Constraint Programming (CD-DP'15)*, pages 61–69, 2015.
- [73] Kenneth J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6 :93–94, 1987.
- [74] Ewa Kubicka and Allen J Schwenk. An introduction to chromatic sums. In *Proceedings of the 17th conference on ACM Annual Computer Science Conference*, pages 39–45. ACM, 1989.
- [75] Qinghua Wu and Jin-Kao Hao. An effective heuristic algorithm for sum coloring of graphs. *Computers & Operations Research*, 39(7) :1593 – 1600, 2012.
- [76] Una Benlic and Jin-Kao Hao. A study of breakout local search for the minimum sum coloring problem. In *Simulated Evolution and Learning*, pages 128–137. Springer, 2012.
- [77] Anders Helmar and Marco Chiarandini. A local search heuristic for chromatic sum, 2011.
- [78] Yan Jin, Jin-Kao Hao, and Jean-Philippe Hamiez. A memetic algorithm for the minimum sum coloring problem. *Computers & Operations Research*, 43 :318–327, 2014.
- [79] Yan Jin and Jin-Kao Hao. Hybrid evolutionary search for the minimum sum coloring problem of graphs. *Information Sciences*, 352 :15–34, 2016.
- [80] Fred W. Glover, Darwin Klingman, and Nancy V. Phillips. A new polynomially bounded shortest path algorithm. *Operations Research*, 33(1) :65–73, 1985.
- [81] Pablo Moscato and Carlos Cotta. A gentle introduction to memetic algorithms. 2003.
- [82] Zhipeng Lü and Jin-Kao Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1) :241–250, 2010.
- [83] I. M. Ali, S. M. Elsayed, T. Ray, and R. A. Sarker. Memetic algorithm for solving resource constrained project scheduling problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2761–2767, May 2015.
- [84] Christian Prins, Caroline Prodhon, and Roberto Wolfler Calvo. A memetic algorithm with population management (ma—pm) for the capacitated location-routing problem. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 183–194, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [85] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *J. Comb. Optim.*, 3(4) :379–397, 1999.

- [86] Zbigniew Kokosiński and Krzysztof Kwarciany. On sum coloring of graphs with parallel genetic algorithms. In Bartłomiej Beliczynski, Andrzej Dzielinski, Marcin Iwanowski, and Bernardete Ribeiro, editors, *Adaptive and Natural Computing Algorithms*, pages 211–219, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [87] Olfa Harrabi, Joughaina Chaouachi Siala, and Hend Bouziri. On integrating an iterated variable neighborhood search within a bi-objective genetic algorithm : Sum coloring of graphs case application. *Electronic Notes in Discrete Mathematics*, 66 :55 – 62, 2018. 5th International Conference on Variable Neighborhood Search.
- [88] Qinghua Wu, Qing Zhou, Yan Jin, and Jin-Kao Hao. Minimum sum coloring for large graphs with extraction and backward expansion search. *Applied Soft Computing*, 62 :1056 – 1065, 2018.
- [89] Weibo Lin, Mingyu Xiao, Yi Zhou, and Zhenyu Guo. Computing lower bounds for minimum sum coloring and optimum cost chromatic partition. *Computers & Operations Research*, 109 :263 – 272, 2019.
- [90] Carlos Ansótegui, Joel Gabàs, Yuri Malitsky, and Meinolf Sellmann. Maxsat by improved instance-specific algorithm configuration. *Artificial Intelligence*, 235 :26 – 39, 2016.
- [91] J. Argelich, C. M. Li, F. Manyà, and Z. Zhu. Many-valued minsat solving. In *2014 IEEE 44th International Symposium on Multiple-Valued Logic*, pages 32–37, May 2014.
- [92] Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. The minimum satisfiability problem. *SIAM J. Discrete Math.*, 7(2) :275–283, 1994.
- [93] Flavia Bonomo and Mario Valencia-Pabon. Minimum sum coloring of  $P_4$ -sparse graphs. *Electronic Notes in Discrete Mathematics*, 35 :293–298, 2009.
- [94] Flavia Bonomo and Mario Valencia-Pabon. On the minimum sum coloring of  $P_4$ -sparse graphs. *Graphs and Combinatorics*, 30(2) :303–314, 2014.
- [95] Hossein Hajiabolhassan, Mojtaba L Mehrabadi, and Ruzbeh Tusserkani. Minimal coloring and strength of graphs. *Discrete Mathematics*, 215(1) :265–270, 2000.
- [96] Chu-Min Li, Zhiwen Fang, and Ke Xu. Combining maxsat reasoning and incremental upper bound for the maximum clique problem. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*, pages 939–946, 2013.
- [97] Simon Caillard, Laure Brisoux Devendeville, and Corinne Lucet. A planning problem with resource constraints in health simulation center. In *Optimization of Complex Systems : Theory, Models, Algorithms and Applications, WCGO 2019, World Congress on Global Optimization, Metz, France, 8-10 July, 2019.*, pages 1033–1042, 2019.



## Annexe A

# CURRICULUM VITAE Etendu

### A.1 Etat Civil

**Corinne LUCET**, épouse VASSEUR  
Née le 19/09/1965  
Mariée (4 enfants)  
Grade : Maître de Conférences depuis Septembre 1994  
Hors Classe depuis Septembre 2011  
Section : 27

Adresse : 88, rue Victor Hugo, 80440 Boves, FRANCE  
email : corinne.lucet@u-picardie.fr  
Portable : 06 87 44 69 52

### A.2 Etablissements

**Université de Picardie Jules Verne**

**Unité d'Enseignement**

Institut Universitaire de Technologie  
Département Informatique  
Le Bailly, avenue des Facultés  
80025 Amiens

**Unité de Recherche**

Laboratoire Modélisation Information & Systèmes  
EA 4290  
33 rue Saint Leu  
80039 Amiens

## A.3 Titres et Diplômes

### Thèse de Doctorat 1990-1993

**Doctorat de l'Université de Technologie de Compiègne**

spécialité "Contrôle des Systèmes"

*Méthode de décomposition pour l'évaluation de la fiabilité des réseaux*

Directeur de thèse : Pr. Jacques CARLIER      Université de Technologie de Compiègne  
Rapporteur : Pr. Philippe CHRETIENNE      Université Pierre et Marie Curie  
Rapporteur : Pr. Loÿs THIMONIER      Université de Picardie Jules Verne

### DEA Informatique 1988-1989

**Université de Technologie de Compiègne**

spécialité "Contrôle des Systèmes"

*Problèmes d'affectations de conteneurs pour le transport ferroviaire*

Laboratoire : Centre de Robotique Electrotechnique et Automatique  
Université de Picardie Jules Verne

### Maîtrise d'Informatique 1986-1988

Université de Picardie Jules Verne. Mention Bien.

### DEUG Mathématiques et Physique 1984-1986

Université de Picardie Jules Verne.

### Baccalaureat Série S - 1984

Académie d'Amiens, Lycée M. Michelis, Amiens.

## A.4 Synthèse de carrière

**(1991-1993)** J'ai effectué ma thèse de doctorat au sein du laboratoire HeuDiaSyC de l'Université de Technologie de Compiègne, sous la direction du Professeur Jacques Carlier. Lors de cette thèse, soutenue le 20 décembre 1993, j'ai étudié et développé une méthode basée sur la décomposition arborescente des graphes, pour la résolution du problème NP-difficile de la fiabilité des réseaux. Cette thèse fut d'abord financée par une allocation délivrée par le Conseil Régional de Picardie. Lors de ma dernière année, j'occupais un poste d'Attaché Temporaire d'Enseignement et de Recherche au département Informatique de l'Institut Universitaire de Technologie d'Amiens.

**(1993-1999)** A l'issue de ce contrat, j'ai été recrutée par la commission de spécialistes en 27ème section sur un poste de Maître de Conférences à l'IUT d'Amiens. Je commençais alors l'enseignement de l'algorithmique et de la programmation en langage Pascal, puis en langage C pour les étudiants de première année (L1). J'ai ensuite pris en charge le module de Recherche Opérationnelle, alors dispensé à l'ensemble des étudiants de deuxième année (L2). Rapidement après ma nomination, la direction du département informatique m'a proposé la responsabilité des stages des étudiants de deuxième année, fonction que j'ai assurée jusqu'en juin 2011. Celle-ci consistait à accompagner les étudiants tout au long de leur démarche de recherche de stage, à valider chacun des sujets proposés par les entreprises, à accompagner également le cas échéant les entreprises dans leur formulation de proposition de stage, et enfin à organiser les suivis de stages par les enseignants, ainsi que le déroulement de l'ensemble des soutenances.

Pour la recherche, je suis restée attachée au laboratoire HeuDiaSyC, dans l'équipe Réseaux et Optimisation jusqu'en 1999. J'y ai travaillé en collaboration avec le Professeur Jacques Carlier sur des problèmes de fiabilité de réseaux. Dès 1995, j'ai pu encadrer la thèse de doctorat de M. Jean-François Manouvrier, intitulée "Méthodes de décomposition pour résoudre des problèmes combinatoires sur les graphes" et soutenue en Novembre 1998. Cette thèse traitait principalement d'une extension de la fonction de connexité de réseaux (la 2-arête connexité) dont les composants sont susceptibles de défaillance, par une décomposition arborescente du graphe.

**(1999-2008)** C'est en 1999 que j'ai intégré le tout jeune laboratoire de recherche en informatique d'Amiens (LaRIA, FRE 2733) dirigé par le Professeur Patrice Seebold et plus particulièrement l'équipe Graphe et Optimisation Combinatoire. J'ai maintenu une collaboration étroite avec le Professeur Jacques Carlier et M. Aziz Moukrim du laboratoire HeuDiaSyC (UTC). Cette collaboration a été alimentée tout au long de ces années par des projets de recherche soutenus et financés par le Conseil Régional de Picardie. Nous avons alors co-encadré de 2001 à 2005 la thèse de Mme Florence Mendes (initialement étudiante en Master 2 à Amiens), financée par le Conseil Régional de Picardie (projet 2000.3) dont la thématique principale était l'emploi de méthodes exactes (décomposition arborescente - décomposition linéaire) pour un problème central en optimisation combinatoire : la coloration de graphes. A travers ce projet nous avons travaillé également sur des problèmes de planification d'emploi du temps pour l'entreprise CORIOLIS Service située à Amiens.

En 2004, le Professeur Nikolaos Limnios, du Laboratoire de Mathématiques Appliquées de Compiègne (LMAC) m'a proposé une collaboration sur l'emploi des Diagrammes de Décision Binaires (BDD) pour le calcul de la fiabilité des réseaux. Notre projet a été soutenu par le Conseil Régional de Picardie (projet DIVA 2004.2), qui a permis le financement de la thèse de M. Gary Hardy de 2004 à 2007, sur cette thématique. Nous avons obtenus des résultats très intéressants mettant en relation certaines caractéristiques du graphe représentant le réseau et la taille des BDD.

De 2000 à 2009, je suis intervenue de manière ponctuelle et en parallèle de mes enseignements à l'IUT, en Master 1&2 de l'UFR des Sciences d'Amiens, en collaboration avec Mme Yu Li (LaRIA/UPJV). Nos cours, proposés comme modules de recherche, portaient sur une initiation à la programmation dynamique, aux techniques de branch-and-bound et aux algorithmes génétiques. J'étais responsable des cours présentiels "Modélisation, optimisation, complexité et algorithmes" (MOCA B1) jusqu'en 2005, et ensuite du module "Recherche opérationnelle et aide à la décision" (RCP101) jusqu'en 2009, pour la formation ingénieur du CNAM Picardie. De 2008 à 2011 j'ai également enseigné la théorie des langages en L2 au département informatique de l'IUT.



**(2008-2014)** Cette année 2008 fut l'année de la création du laboratoire Modélisation, Information & Systèmes (MIS EA 4290), duquel je suis membre. Il résulte de la fusion de deux laboratoires de l'Université de Picardie Jules Verne : Le Laboratoire de Recherche en Informatique d'Amiens (LaRIA, FRE 2733) et le Centre de Robotique, d'Électrotechnique et d'Automatique (CREA, EA 3299).

Suite à nos précédentes coopérations, nous avons entrepris avec M. Moukrim et Mme Li, l'étude d'un nouveau problème de coloration, le problème de la somme chromatique d'un graphe. Pour ce problème, nous avons choisi de développer des méthodes heuristiques et méta-heuristiques. Ces travaux ont également été financés par le Conseil Régional de Picardie de 2007 à 2010 (projet SCOOP) qui a attribué une allocation de recherche pour la thèse de Mme Kaoutar Sghiouer (soutenue en mars 2011). A travers ce projet nous avons également entamé une collaboration avec les Professeurs Enmin Song (2008) et Renchao JING (2009) de l'Université des Sciences et Technologie de Huazhong, Wuhan, en Chine. Ils furent tous deux accueillis durant un mois comme professeur invité de l'UPJV, au laboratoire MIS.

Les premières approches du problème de planification de personnels pour CORIOLIS Services, nous ont donné une certaine visibilité au sein de nos laboratoires relativement à cette thématique de recherche, et nous ont permis d'obtenir un contrat CIFRE avec l'entreprise ICTS France, pour le financement de la thèse de M. Anas Abdoul Soukour (2009-2012). Ce travail fut un début de collaboration avec Mme Laure Brisoux-Devendeville (MIS/UPJV) et une poursuite de mes collaborations avec M. Aziz Moukrim (Heudiasyc/UTC). Nous avons, en parallèle, obtenu le financement du projet PRIMA (2011-2014) par le Conseil Régional de Picardie, pour l'étude de problèmes de localisation de ressources et de routage de véhicules en milieu urbain.

J'ai obtenu la promotion Hors-Classe en juin 2011. J'ai dû cesser de travailler de septembre 2011 à Septembre 2013 à cause d'un problème de santé et j'ai repris le travail en temps partiel thérapeutique durant l'année universitaire 2013-2014.

**(2014-2019)** Au début de l'année 2014 j'ai entamé une collaboration avec M. Chumin Li (Pr - MIS/UPJV) sur une approche SAT pour la résolution du problème de la somme chromatique d'un graphe. Grâce à une allocation ministérielle, obtenue en octobre 2014, nous avons co-encadré la thèse de M. Clément Lecat (soutenue en Novembre 2017), et avons obtenus des résultats marquants. M. Lecat a pour ces travaux obtenu le prix du jeune chercheur de la ROADEF en 2017. Dans ce même temps, avec Mme Brisoux-Devendeville (MIS/UPJV) nous avons travaillé pour le centre de pédagogie active SimuSanté (Centre Hospitalier à Amiens) sur un projet de recherche sur la planification de tâches sous contraintes de ressources. Cette première collaboration sous contrat à déboucher en septembre 2017 sur un financement de thèse (encadrement Mme Brisoux-Devendeville - cofinancement Région Hauts-de-France/SimuSanté) et une poursuite de nos collaborations jusqu'en 2020.

Depuis novembre 2018, dans le cadre du projet LORH, avec Laure Brisoux et Gilles Dequen (MIS/UPJV) nous travaillons sur un autre problème d'optimisation de planification, en collaboration avec l'entreprise EVOLUCARE. Il s'agit pour ce projet d'optimiser le parcours des patients en milieu hospitalier. Une allocation CIFRE dont bénéficie Olivier Gérard (Doctorant UPJV) accompagne ce projet.

## A.5 Activités Scientifiques

### A.5.1 Présentation des thématiques de recherche

**Mots-Clés :** Résolution de problèmes NP-difficiles, Graphes, Décomposition arborescente et linéaire, Diagramme de Décision Binaire, Heuristiques et Métaheuristiques, Coloration de graphes, Fiabilité des réseaux, Localisation/Routage, Planification de tâches.

Mes activités de recherche s'effectuent au sein de l'équipe Graphe, Optimisation et Contraintes (GOC) du laboratoire Modélisation Information & Systèmes (MIS). Elles se concentrent essentiellement sur l'étude d'algorithmes pour la résolution de problèmes combinatoires modélisés sous la forme de graphes. Les problèmes étudiés sont des problèmes de fiabilité de réseaux, des problèmes de coloration de graphe, des problèmes de tournées de véhicules et enfin des problèmes de planification de tâches sous contrainte de ressources. Ces problèmes sont tous NP-difficiles. Le temps nécessaire à l'obtention d'une solution optimale, est toujours une fonction exponentielle de la taille du problème, qui dans le cas de problèmes de graphes est soit le nombre de sommets et/ou le nombre d'arêtes. Pour les résoudre deux approches sont envisageables. La première consiste à déterminer une approximation de la solution optimale du problème à l'aide d'algorithmes gloutons, d'heuristiques et/ou de métaheuristiques. La seconde consiste à développer une méthode exacte qui calcule la ou les solutions optimales du problème. Lors de mes travaux j'ai pu aborder chacun des problèmes ci-dessus cités par l'une et l'autre de ces approches. Classiquement, les méthodes exactes sont proposées pour des instances de petites tailles. Ainsi, une solution optimale peut-être obtenue et éventuellement servir d'étalon pour l'évaluation des algorithmes approchés qui sont généralement employés pour les instances de grande taille. Seuls les problèmes de fiabilité de réseaux ont été uniquement traités par des méthodes exactes, comme la décomposition arborescente d'un graphe et les diagrammes de décisions binaires (BDD pour Binary Decision Diagram). Pour les trois autres problèmes, nous avons chaque fois proposé une résolution exacte et approchée. Une résolution exacte par des paradigmes très différents comme la programmation dynamique (décomposition arborescente), la programmation linéaire, des algorithmes de branch-and-bound et la formulation SAT. Une résolution approchée à l'aide d'algorithmes gloutons, de métaheuristiques (algorithme génétique, recherche tabou, recuit simulé), et de recherche locale. Très récemment et en collaboration avec Clément Lecat et Chumin Li, nous avons proposé une méthode originale basée sur une représentation de l'ensemble des solutions sous forme de motifs d'entiers. Ces travaux nous ont permis de calculer des bornes supérieures pour la force d'un graphe et des bornes inférieures pour la somme coloration. Le schéma de la Figure A.1 présente de manière concise, les différents problèmes que j'ai abordés durant ma carrière scientifique, et les différentes méthodes utilisées pour leur résolution.

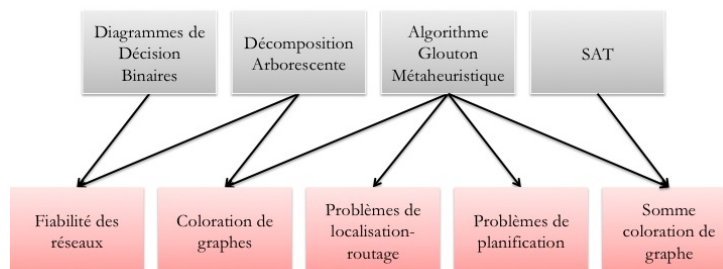


FIGURE A.1 – Méthodes & Problèmes combinatoires étudiés

### A.5.2 Problèmes de Fiabilité des réseaux

Le problème de la fiabilité d'un réseau consiste à calculer la probabilité que sa fonction de communication entre différents sites soit assurée, sachant que ses composants, les sites et/ou les

liaisons, sont sujets à des défaillances. Le réseau est modélisé par un graphe stochastique, ou chaque site du réseau est représenté par un sommet du graphe, chaque lien entre deux sites par une arête et une probabilité de fonctionnement est associée à chacun des composants du graphe. Pour le problème de fiabilité, on associe aux sommets et/ou aux arêtes une probabilité, qui représente la probabilité que le composant soit en état de fonctionnement à un instant donné  $t$ . Il existe différentes fonctions de communication pour les réseaux. Par exemple la connexité tous-terminaux exigera que tous les sites du réseau soit reliés, c'est à dire qu'il existe un chemin de composants en état de fonctionnement reliant tout couple de sites. La connexité K-terminaux demande que tous les sites d'un sous ensemble donné K soient reliés.

Ces problèmes sont NP-difficiles, il n'existe donc pas d'algorithmes efficaces permettant de calculer la fiabilité d'un réseau dans le cas général. Lors de l'étude de la résolution de ces problèmes, je me suis intéressée à deux types de méthodes, la *décomposition arborescente* d'un graphe en collaboration avec le Professeur Jacques Carlier (Heudiasyc/UTC) et M. Jean-François Manouvrier (doctorant UTC) et les *diagrammes de décision binaire* en collaboration avec le Professeur Nikolaos Limnios du Laboratoire de Mathématiques de Compiègne (LMAC/UTC), et M. Gary Hardy (doctorant UPJV).

**Décomposition arborescente de graphe** La méthode de décomposition arborescente d'un graphe est une méthode généraliste, permettant de résoudre de manière exacte certains problèmes NP-difficiles pour des familles de graphes spécifiques, appelées les "k-arbres partiels". Elle consiste à partitionner le graphe en sous-ensembles de sommets, les parties, de telle sorte que la structure du problème soit préservée. Les parties de la partition ainsi obtenue sont reliées les unes aux autres relativement aux arêtes du graphe, pour former un "arbre de parties", d'où la dénomination de décomposition arborescente. La résolution du problème sur le graphe doit pouvoir se calculer à partir des solutions partielles du problème pour les différentes parties de la partition. Pour les problèmes NP-difficiles, la complexité temporelle des méthodes de résolution classiques est généralement une fonction exponentielle de la taille du graphe, c'est-à-dire, son nombre de sommets et/ou son nombre d'arêtes. Dans le cas d'une résolution par la méthode de décomposition arborescente, la complexité temporelle est une fonction exponentielle de la taille maximum des parties, ce qui dans le cas des k-arbres partiels est une valeur bornée par k. Il est donc possible lorsque cette valeur k est petite de fournir une solution optimale en un temps polynomial en fonction de la taille du graphe. Pour une décomposition du graphe donné, la cardinalité maximale des parties qui la composent définit sa "largeur-arbre".

Tous les problèmes de graphes ne peuvent être résolus par une méthode de décomposition arborescente. De nombreux auteurs, comme Courcelle (1990), Boadlander(1995) et Arnborg (1989) ont classifié les problèmes de graphes pour identifier ceux potentiellement solvables par cette méthode. Ils ont établis par exemple que tous les problèmes formulables en logique du second-ordre monadique sont des problèmes à états finis et que tous les problèmes à états finis peuvent être résolus par une décomposition arborescente. Mais lorsqu'un problème se pose, il ne suffit pas de constater qu'il appartient à cette classe, faut-il encore déterminer une décomposition arborescente de largeur-arbre k. Or, pour un graphe G donné, et une valeur k donnée, répondre à la question " la largeur-arbre de G est-elle inférieure ou égale à k" est un problème NP-Complet (Arnborg 1987).

Pour le problème de la fiabilité des réseaux, bien que le principe de la méthode de décomposition arborescente du graphe stochastique fut énoncé par Rosenthal en 1977, aucune expérimentation permettant de mettre en évidence l'efficacité de cette technique sur des réseaux réels n'existait. La difficulté de trouver une décomposition arborescente et la complexité de la recombinaison des solutions partielles, nous a conduit à développer une variante de la décomposition arborescente : la décomposition linéaire d'un graphe. L'arbre des parties est réduit à un chemin. On parle alors de la largeur-linéaire du graphe. Même si le problème d'optimisation associé reste NP-difficile, l'implémentation est plus simple et la construction heuristique de la décomposition se fait à partir d'une numérotation des sommets du graphe. Nous avons proposé plusieurs heuristiques comme le

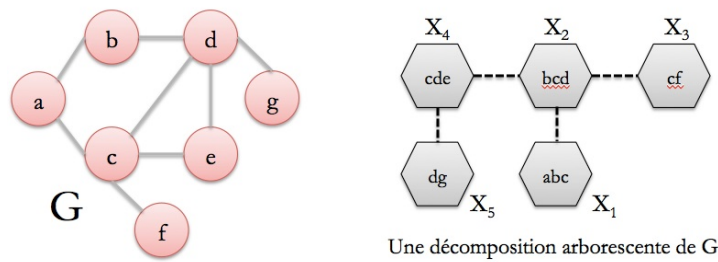


FIGURE A.2 – Décomposition arborescente d'un graphe.  
La combinaison des solutions partielles des  $X_i$  fournira la solution globale du problème

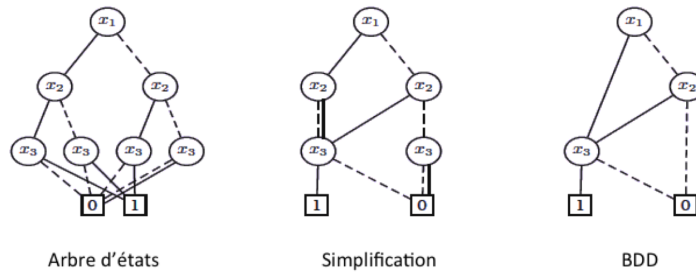
recuit simulé ou la numérotation dynamique pour produire une décomposition linéaire de largeur réduite.

Nous avons ensuite proposé un ensemble d'algorithmes permettant de tester différentes fonctions de connexité sur les réseaux : connexité tous-terminaux,  $K$ -terminaux, 2-connexité et 2-arêtes-connexité. Notre modélisation s'appuie sur la modélisation proposée par Arnborg (1987), que nous avons étendue pour pouvoir représenter des fonctions plus complexes comme la 2-connexité et la 2-arêtes-connexité. Cette modélisation repose sur la définition de classes d'équivalence de sous-structures connexes du graphe (les solutions partielles). Ces classes sont représentées sous la forme de partitions des sommets. À chacune de ces classes est associée la probabilité d'occurrence de la structure connexe décrite par la partition. La fiabilité du réseau est une combinaison des probabilités des solutions partielles, les partitions. Or, ce nombre de partitions croît de manière super-exponentielle en fonction du nombre de sommets dans chacune des parties. Il correspond en effet aux nombres de Stirling de seconde espèce. C'est cette même modélisation par partitions des sommets que nous retrouverons dans nos travaux sur la coloration de graphe. Nous avons pu résoudre ces problèmes pour des familles de graphes dont la largeur-linéaire n'excédait pas  $k=15$ , mais pour des graphes de grande taille (plus de 1000 nœuds).

Ces travaux sur les problèmes de fiabilité de réseaux ont donné lieu à deux publications de très bonne audience dans les revues *Discrete Applied Mathematics* [1] et *Algorithmica* [2], des conférences internationales et nationales.

**Diagrammes de Décision Binaires** Suite aux résultats très encourageants obtenus par la décomposition linéaire pour le calcul de la fiabilité des réseaux, nous avons étudié en collaboration avec le Professeur Nikolaos Limnios du Laboratoire de Mathématiques Appliquées de Compiègne (LMAC/UTC), l'emploi des puissantes structures que sont les Diagrammes de Décision Binaires (BDD). Les BDD ont été introduits par Akers en 1978 pour représenter les circuits électriques. L'utilisation des BDD dans le domaine de la fiabilité a été introduite par Madre, Coudert et Rauzy au début des années 90. Ces travaux ont permis, pour la première fois, de traiter qualitativement et quantitativement de grands arbres de défaillance en quelques secondes. Nous avons donc souhaité tester l'intérêt des BDD pour l'évaluation exacte de la fiabilité  $K$ -terminaux de réseau. Notre contribution principale consistait en l'utilisation de la modélisation des solutions partielles par partitions, couplée à la puissance de représentation des BDD. Comme le montre la Figure A.3, une fonction logique est représentée de manière très concise par un BDD.

Nos travaux se sont déroulés dans le cadre d'un projet de recherche, supporté et financé par le Conseil Régional de Picardie (2004-2007), qui a permis de financer également la thèse de M. Gary Hardy (doctorant UPJV). Nous avons proposé un algorithme permettant de calculer la fiabilité  $K$ -terminaux, s'appuyant sur la représentation de la fonction de structure de réseau par les BDD. Contrairement aux autres méthodes également basées sur les BDD, notre algorithme a une complexité en temps indépendante de la taille de l'ensemble  $K$  des sommets terminaux. La complexité ne dépend uniquement que de la largeur-linéaire du réseau (donc du graphe comme dans la décomposition linéaire). Les résultats expérimentaux obtenus sur différents benchmarks de la littérature prouvent l'efficacité de cette approche. Pour des largeurs linéaires d'ordre 4 ou 5, des réseaux de plusieurs dizaines de milliers de nœuds peuvent être traités. Nous nous sommes aussi

FIGURE A.3 – Représentation canonique de  $f = (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$  par un BDD

intéressés aux problèmes de synthèse de réseaux. Dans ce type de problème, l'objectif consiste à déterminer une topologie de réseau le plus généralement la moins coûteuse possible et satisfaisant un critère de fiabilité minimale. Ces problèmes sont eux aussi NP-difficiles. Nous avons proposé une heuristique, toujours basée sur les BDD pour l'évaluation des solutions finales. Les très bons résultats obtenus par rapport aux meilleurs algorithmes de la littérature, prouvent que les BDD sont des outils très puissants pour l'aide à la synthèse de réseaux. Ces travaux ont donné lieu à deux publications dans des journaux internationaux de haut rang, IEEE Trans. Reliability [3] et Performance Evaluation Review [12], dans des conférences internationales et nationales.

### A.5.3 Problèmes de coloration de graphes

#### Nombre chromatique d'un graphe

Toujours en résolution exacte, et dans l'optique de l'utilisation de la méthode de décomposition arborescente, nous avons travaillé sur un problème central en Optimisation Combinatoire, le problème NP-difficile de coloration de graphe. De nombreux problèmes difficiles comme l'allocation de fréquences ou les problèmes d'emploi du temps peuvent se modéliser par un problème de coloration de graphe. Les sommets représentent les entités d'un système et deux sommets sont liés par une arête si les deux entités correspondantes sont en relation (incompatibles, voisins, etc.). Afin de définir des classes d'entités non liées deux à deux, c'est-à-dire indépendantes, pour éventuellement leur appliquer un traitement quelconques (les stocker, les programmer, etc.), on associe à chaque sommet une "couleur". Deux sommets liés par une arête ne doivent pas se voir attribuer une même couleur, ils ne peuvent être traités ensemble. Deux sommets partageant la même couleur sont indépendants. Le but est de trouver le nombre chromatique du graphe, c'est-à-dire le nombre minimum de couleurs nécessaires pour colorier le graphe, tout en respectant les contraintes de voisinage.

Lors de la thèse de Mme Florence Mendes, nous avons établi des propriétés de pré-traitements du graphe permettant de réduire sa taille tout en maintenant ses caractéristiques au regard de son nombre chromatique. Nous avons également introduit des techniques de branch-and-bound dans la méthode de décomposition afin de traiter les graphes tests de la littérature et nous mesurer aux meilleures méthodes publiées. Ces travaux ont été menés en collaboration avec le Professeur Jacques Carlier et le Professeur Aziz Moukrim du Laboratoire HeuDiaSyC de Compiègne (UTC), dans le cadre d'un projet financé par le Conseil Régional de Picardie. Était également attachée à ce projet, la thèse de Florence Mendes (2005), actuellement Maître de Conférence à l'Université de Bourgogne. Ces travaux ont été publiés dans le journal international Computers & Operations Research [49], dans des conférences internationales et nationales.

#### Somme chromatique et force d'un graphe

A la fin de nos travaux sur la coloration de graphe, nous nous sommes intéressés à l'élaboration de méthodes approchées comme les algorithmes gloutons, les heuristiques et les méta-heuristiques. Nous avons principalement étudié une extension du problème classique de la coloration de graphe, le problème de la *somme coloration* d'un graphe (MSCP).

Pour ce problème de somme coloration, on associe à chaque couleur un poids qui peut par exemple représenter une date dans le cas d'un ordonnancement de processus, ou un coût d'utilisation dans le cas d'allocation de fréquences. Le but n'est plus alors de trouver une coloration minimisant le nombre de couleurs, mais de trouver une coloration minimisant la somme des poids des couleurs utilisées. Pour un graphe  $G$  donné, cette somme est appelée la *somme chromatique* de  $G$  et elle est notée  $\Sigma(G)$ . Cette somme correspondrait au temps moyen d'achèvement pour l'ordonnancement de tâches, et au coût global pour l'allocation de fréquences. Ce problème est un problème NP-difficile, introduit par Kubika en 1998 et beaucoup moins étudié dans la littérature que le problème du nombre chromatique. Le nombre minimum de couleurs utilisées dans une solution optimale de MSCP est appelé la *force* du graphe, notée  $s(G)$ . La particularité de MSCP est que cette force peut-être arbitrairement plus grande que le nombre chromatique du graphe. L'espace des solutions est donc difficile à explorer, car difficile à borner.

**Méthodes approchées :** Les travaux que nous avons mené sur ce problème ont également été soutenus par le Conseil Régional de Picardie dans le cadre du projet SCOOP en collaboration avec le Professeur Aziz Moukrim (Heudiasyc/UTC) et Mme Yu Li. Une allocation de recherche à ainsi permis à Mme Kaoutar Sghiouer de préparer une thèse (UTC) soutenue en Mars 2011. Nous avons dans un premier temps proposé deux algorithmes gloutons MDSATUR et MRLF, basés sur les algorithmes gloutons de référence en coloration de graphe, DSATUR de Brelaz (1979) et RLF de Glover (1985). Une méthode de recherche locale de type Destruction/Construction a également été proposée, elle permet d'intensifier la fouille de l'espace de recherche autour d'une solution. Celle-ci a été combinée à un algorithme évolutionnaire, pour donner l'algorithme MA-MSCP qui nous a permis d'obtenir des bornes supérieures de la somme chromatique pour l'ensemble des benchmarks de la littérature. En utilisant ce même algorithme sur le graphe complémentaire, nous avons également déterminé des bornes inférieures de la somme.

**Formulation SAT :** Le problème de satisfaisabilité booléenne (SAT) est un problème de décision dont l'objectif est de déterminer une instanciation d'un ensemble de variables booléennes, appelé interprétation afin de satisfaire une formule sous forme normale conjonctive (CNF). Par exemple la fonction  $F = (a \vee c \vee \neg d) \wedge (b \vee \neg c \vee d)$  est satisfaite par l'interprétation suivante :  $(a = \text{vrai}, b = \text{vrai})$ .  $(a \vee c \vee \neg d)$  est appelée une clause. Peuvent être formulés sous cette forme, les problèmes pour lesquels les solutions sont "oui" ou "non", c'est à dire existe t-il une interprétation des variables qui satisfasse la fonction  $F$ .

Le problème de la somme coloration étant un problème d'optimisation nous avons donc utilisé *Weighted Partial MaxSAT* qui introduit la notion de clauses dures et de clauses souples pondérées afin de modéliser les problèmes d'optimisation. L'objectif est alors de déterminer une interprétation satisfaisant l'ensemble des clauses dures (comme pour SAT) tout en maximisant la somme des poids des clauses souples satisfaites. Une formulation duale, appelée *Weighted Partial MinSAT* aura pour objectif de minimiser la somme des poids des clauses souples non-satisfaites.

L'étude de la résolution du problème de la somme coloration par ce type de formulation, et donc par des solveurs SAT puissants, fut le centre d'une première collaboration avec le Professeur Chumin Li (MIS/UPJV), concrétisée par un co-encadrement de la thèse de Clément Lecat, financée par une allocation ministérielle. Lors de cette thèse, nous avons proposé plusieurs modélisations de MSCP sous forme de problèmes Weighted Partial MaxSAT et de problèmes Weighted Partial MinSAT. Pour ces deux formalisations, le nombre de clauses est dépendant du nombre de sommets et d'arêtes du graphe, mais également du nombre de couleurs que nous engageons pour définir l'ensemble des solutions à explorer. Pour chaque sommet  $v$  du graphe, et pour chaque couleur  $c$  engagée, une variable  $x_{v,c}$  est générée. Nous avons montré que pour chaque sommet, le nombre de couleurs à considérer est  $\min\{s(G), d(v) + 1\}$ . L'impact de cette restriction sur la taille des instances Weighted Partial MaxSAT générées à partir des benchmarks COLOR et DIMACS de la littérature est très concluant. Ces travaux ont fait l'objet de publications dans une conférence internationale (International Conference on Principles and Practice of Constraint Programming (CP-2015) [72] et dans des conférences nationales.

**Représentation de l'espace des solutions par Motifs :** Bien que les résultats obtenus

pour MSCP avec la modélisation SAT soient intéressants, reste le problème de la valeur de la force  $s(G)$ , dont la détermination est lui même un problème NP-difficile dans le cas général. La solution dans de tel cas consiste à utiliser une borne supérieure de bonne qualité. Or, seules quelques bornes relatives au degré du graphe étaient connues au début de nos travaux.

Grâce à notre expérience de la décomposition arborescente de graphe, et à la modélisation des solutions sous forme de partitions des sommets du graphe, nous avons proposé une toute nouvelle vision du problème MSCP. En effet, ce qui importe avant tout pour le problème de la somme coloration est la cardinalité des parties. Ces cardinalités sont des entiers. Nous avons donc défini la notion de *motifs*, qui sont des séquences ordonnées d'entiers, permettant de symboliser l'espace des solutions de manière agrégée. La relation d'ordre définie sur l'ensemble de ces motifs nous a permis de mettre en avant une nouvelle borne inférieure de la somme chromatique  $\Sigma(G)$  et une nouvelle borne supérieure de la force  $s(G)$ . Cette dernière a surpassé les résultats proposés dans la littérature. Ces travaux ont été publiés dans une revue d'audience internationale "Discrete Applied Mathematics" [7] et dans une conférence internationale de haut rang "Conference on Artificial Intelligence (AAAI-17)" [8], et ont permis à Clément Lecat d'obtenir le prix du jeune chercheur de la ROADEF en 2017.

#### A.5.4 Problèmes Logistiques

Nous avons également élargi notre domaine d'application en travaillant sous forme de projet de recherche en collaboration avec le laboratoire HeudiaSyC, sur des problèmes de planification d'emploi du temps et des problèmes de transport (localisation/routage).

##### Localisation-Routage

Le problème de localisation/routage provient de l'idée de combiner deux niveaux de décision de logistique : la localisation de dépôts et l'élaboration de tournées de véhicules. Bien que l'énoncé de ce problème remonte aux années 60, c'est à partir des années 90 qu'il a été mis en évidence d'intégrer les tournées de véhicules lors de la phase de localisation des dépôts. L'imbrication des deux problèmes rend la résolution plus difficile, mais est nécessaire pour qu'une solution optimale puisse être atteinte. Les travaux effectués sur ce problème le furent dans le cadre d'un projet (PRIMA) financé par le Conseil Régional de Picardie, en collaboration avec M. Aziz Moukrim et Mme Laure Brisoux Devendeville. Nous avons obtenu une allocation de recherche qui a permis à M. Sohaib Afifi d'effectuer une thèse (UTC) soutenue en octobre 2014. La formulation du problème sous forme de Programme Linéaire en Nombres Entiers nous a permis par l'utilisation du solveur CPLEX de vérifier les coûts optimaux des plus petites instances de ces benchmarks, afin d'évaluer l'efficacité des méthodes heuristiques développées par les deux équipes HeuDiaSyc et MIS. Cette phase est généralement riche d'enseignement sur la structure même du problème et nous apporte donc les éléments de caractérisation des solutions nécessaires à l'élaboration des métaheuristiques. Ces métaheuristiques s'appliquent à toutes sortes de problèmes discrets, comme notre problème de localisation-routage, et contiennent toujours une partie stochastique permettant de faire face à l'explosion combinatoire du nombre de solutions possibles. Elles sont souvent inspirées de méthodes d'optimisation (pas forcément discrète) de domaines extérieurs, comme la physique, la biologie ou encore l'éthologie. Nous avons développé des algorithmes évolutionnaires, basés sur des modèles de reproduction et d'évolution de populations, telles que les algorithmes génétiques ou mémétiques. Pour ces méthodes les solutions sont croisées entre elles, d'itération en itération afin de donner en héritage aux générations suivantes les meilleures caractéristiques. La méthode Cross-Entropy, développée pour la résolution de LRP, pourrait s'apparenter à de telles méthodes, mais cette fois les gènes des meilleures solutions sont mis en commun pour une génération stochastique de nouvelles solutions. La méthode de Cross-Entropy (CE) a été initialement conçue pour estimer la probabilité d'événements rares dans les réseaux stochastiques complexes (Rubinstein 1997) et a fait l'objet d'une adaptation aux problèmes d'optimisation combinatoires difficiles. La méthode CE utilise une formulation des problèmes sous forme de graphe. Schématiquement une matrice de probabilités représente la mémoire du système et les nouvelles solutions sont produites par un processus de sélection de chaînes de Markov. L'algorithme altère la matrice de transition et amplifie les probabilités dans les chaînes de Markov par échantillonnage au cours des itérations.

### Planification

Lors du premier projet concernant le problème de coloration de graphe, nous nous sommes intéressés aux applications possibles, directes ou indirectes de la méthodologie développée. Une première collaboration a commencé avec l'entreprise Coriolis Service (Centre d'Appels à Amiens) sur un problème de calcul d'emploi du temps de personnels, sous contrainte de charge. Cette collaboration a donné lieu à un stage de Master ainsi qu'à l'intervention de Florence Mendes alors Doctorante à l'UPJV. Les caractéristiques de ce problème étaient l'intégration de la multi-compétence des agents. Nous avons également collaboré avec un établissement d'enseignement Amiénois toujours sur des problèmes de calcul d'emploi du temps, concrétisé par un stage ingénieur du CNAM de 6 mois.

#### Projet PLAN-AIR

En continuité de ce thème de septembre 2009 à mars 2012, nous avons co-encadré avec Laure Devendeville (MIS/UPJV), et Aziz Moukrim (HeuDiaSyC/UTC) la thèse d'Anas Abdoul Soukour en contrat CIFRE, sur la planification des personnels dans la sûreté aérienne. Ce projet était une collaboration avec l'entreprise ICTS France, spécialisée dans la sûreté aérienne. ICTS se doit de répondre aux attentes de leurs clients (compagnies aériennes et aéroports) en agents qualifiés, sur différents sites et pour certains créneaux horaires définis par les départs et arrivées des avions. ICTS gère environ 2600 agents de sûreté en France. Le grand nombre de contraintes à considérer et la nature de la demande des clients font de ce problème un problème NP-difficile.

#### Projet SimuStorE

Sujet : Optimisation de la gestion de la ressource appliquée au centre de formation SimUSanté

Avec Laure Brisous-Devendeville, nous travaillons sur le projet SimuStorE depuis Septembre 2016, en collaboration avec le centre de pédagogie active SimUSanté. Celui-ci qui est basé au Centre Hospitalier Universitaire d'Amiens, est le plus grand centre européen de formation et de simulation en santé. A ce titre, ce centre de formation propose et organise un grand nombre de *Parcours pédagogiques* destinés aux étudiants, et à tous types de professionnels de la santé. Ces formations sont souvent des mises en situation qui nécessitent un matériel spécifique allant de l'instrumentation médicale comme des mannequins conçus pour la simulation et la pratique d'une gamme complète de procédures d'évaluation et de soins des malades, mais également des produits pharmaceutiques, des plateformes pédagogiques, des blocs opératoires, des salles équipées de vidéo, ou encore d'un environnement reproduisant les risques domestiques. A la mise à disposition et au maintien de cet ensemble de ressources hétérogène, s'ajoutent les disponibilités des intervenants formateurs et celles des participants. On comprend donc l'enjeu d'une bonne gestion des ressources pour une planification efficace des Parcours pédagogiques du centre SimUSanté.

Ce projet est accompagnée d'une allocation de recherche co-financée par le Conseil Régional des Hauts-de-France et SimUSanté, dont bénéficie Simon Caillard, actuellement en fin de seconde année de thèse au laboratoire MIS/UPJV.

#### Projet LORH

Sujet : Optimisation du parcours patient en milieu hospitalier

Depuis novembre 2018, nous travaillons en collaboration avec l'entreprise EVOLUCARE sur un problème d'optimisation d'emploi du temps. L'objectif de ce projet est de répondre à un ensemble de demandes de rendez-vous, faisant intervenir un ensemble de nombreuses ressources de types différents, et de proposer une solution vérifiant toutes les contraintes associées. La particularité de ces travaux est la généralité des méthodes à mettre en place, afin de répondre à un grand nombre de scénarios possibles de la vie hospitalière. Ce projet est accompagné d'une thèse CIFRE, dont bénéficie Olivier Gérard, doctorant au laboratoire MIS/UPJV.



## A.6 Publications

### A.6.1 Articles de Livre d'audience internationale

- C. Lucet, J.F. Manouvrier, , Exact methods for the network reliability problem, *Mathematical Methods in Reliability*, pages 279-294. Ed. Birkhäuser, 1998
- G. Hardy, C. Lucet N. Limnios., A BDD-based algorithm for computing the K-terminal network reliability, special volume of *Recent Advancement of Stochastic Operations Research*, World Scientific 2007

### A.6.2 Revues d'audience internationale

- J. Carlier, C. Lucet, "A decomposition Algorithm for Network Reliability Evaluation", *Discrete Applied Mathematics* 65 (13) :141-156 (1996)
- C. Lucet, J.F. Manouvrier, J. Carlier, "Evaluating Network reliability and 2-Edge-Connected Reliability in Linear Time for Bounded Pathwidth Graphs" , *Algorithmica* 27(3) : 316-336 (2000)
- C. Lucet, F. Mendes, A. Moukrim, "An Exact Method for Graph Coloring", *Computers & Operations Research*, vol. 33, n° 8, pp. 2189-2207, 2006.
- G. Hardy, C. Lucet N. Limnios, "Probability of connection in regular stochastic networks", *SIGMETRICS Performance Evaluation Review* 34(2) : 9-10 (2006)
- G. Hardy, C. Lucet N. Limnios, "K-terminal Network Reliability Measures with Binary Decision Diagrams", *IEEE Transaction on Reliability*, 56(3) : 506-515 (2007)
- Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, Yu L, "Lower Bounds for the Minimal Sum Coloring Problem". *Electronic Notes in Discrete Mathematics* 36 : 663-670 (2010)
- Abdoul Soukour A., Devendeville L., Lucet C., Moukrim A., "A Memetic Algorithm for staff scheduling problem in airport security service". *Expert Syst. Appl*, 40(18) :7504-7512, (2013).
- Clément Lecat, Corinne Lucet, Chu-Min Li, "Minimum sum coloring problem : Upper bounds for the chromatic strength". *Discrete Applied Mathematics* 233 : 71-82 (2017)

### A.6.3 Conférences internationales

- J. Carlier, C. Lucet, "A decomposition algorithm for network reliability evaluation", *First International Colloquium on Graphs and Optimization (GOI)*, Grimentz, Switzerland, 23-28 August 1992.
- J. Carlier, C. Lucet, "An efficient algorithm for network reliability problems", *Conference of the European Chapter on Combinatorial Optimization, ECCO VI*, Bruxelles, Belgium, April 1993.
- J. Carlier, C. Lucet. "A decomposition method for the connectivity and the biconnectivity problems", *Congrès IFIP 93*, Compiègne, France, July 1993.
- J. Carlier, C. Lucet, "The 2-edge-Reliability Problem", *Conference of the European Chapter on Combinatorial Optimization, ECCO VII*, Milan, Italy, 21-23, Février 1994.

- J.Carlier, C. Lucet , "Computing the reliability of networks with imperfect nodes ", FRAN-CORO, Mons, Belgium, May 1995.
- J. Carlier, C. Lucet, "A decomposition method for the 2-edge-Reliability Problem", Optimisation Days, Centre de Recherche sur les Transports, Montréal, Canada, May 1996.
- J.F. Manouvrier, C. Lucet. "Resolving the network reliability problem with a tree decomposition of the graph", On Principles of Distributed Systems, OPODIS 97, Chantilly, France, December 1997.
- C. Lucet, , "Dynamic Algorithm for Minimal Steiner Tree Problem", International symposium on Combinatorial Optimization, CO'02, Paris, France, April 8-10 2002.
- C. Lucet, F. Mendes, A. Moukrim, "Pre-processings and Linear-decomposition algorithm to solve the k-colorability problem", III Workshop on Efficient and Experimental Algorithm, WEA'2004, LNCS vol. 3054, pages 315-325, Angra dos Reis, Brazil, 25-28 mai 2004.
- G. Hardy, C. Lucet N. Limmios, "Computing all-terminal reliability of stochastic networks with Binary Decision Diagrams", International Symposium on Applied Stochastic Models and Data Analysis (ASMDA 05), Brest , France, 17-20 mai 2005.
- G. Hardy, C. Lucet N. Limmios, "A BDD-based algorithm for computing the K-terminal network reliability", International Workshop on Recent Advances in Stochastic Operations Research, Canmore, Canada August 25-26, 2005.
- G. Hardy, C. Lucet N. Limmios, "Probability of connection in regular stochastic networks", International Symposium on Computer Performance, Modeling, Measurements and Evaluation, PERFORMANCE 2005, Juan-les-Pins, France, october 3-7 2005.
- G. Hardy, C. Lucet N. Limmios "A BDD-based heuristic algorithm for design of reliable networks with minimal cost", International Conference on Mobile Ad Hoc and Sensor Networks (MSN 2006), Hong Kong, China, 13-15 Dec. 2006
- F. Mendes, C. Lucet, A. Moukrim , "Tabu Search to Plan Schedules in a Multiskill Customer Contact Center" , International Conference on Service Systems and Service Management, Vol. 2, pp 1126-1131, Troyes, France, October 2006
- G. Hardy, C. Lucet N. Limmios, "Topological design of communication networks using BDD", Métaheuristiques, Hammamet, Tunisie, 02-04 Novembre 2006
- Y. Li, C. Lucet, A. Moukrim, K. Sghiouer, "Greedy Algorithm for the minimum sum of graph coloring", Workshop International : Logistique et Transport, Mar 2009, Sousse, Tunisie. 2009.
- Y. Li, C. Lucet, A. Moukrim, K. Sghiouer," Lower Bounds for the Minimal Sum Coloring Problem », ISCO 2010, Hammamet, Tunisie, Mars 2010.
- Abdoul-Soukour A., Devendeville L., Lucet C., Moukrim A., "Staff scheduling in airport security service". In INCOM'12 , p. 61-72, Bucarest,Romania, Mai 2012.
- Lecat C., Li C.M., Lucet C., Li Y., "Exact methods for the minimum sum coloring problem". In the doctoral program Proceedings of the 21th International Conference on Principles and Practice of Constraint Programming (CP-2015), Cork, Ireland August 31-September 4, 2015, .p. 61-69, 2015.
- Sohaib Affi , Laure Brisoux , Corinne Lucet , Aziz Moukrim, "Particle Swarm Optimization

for the Location Routing Problem”, In Proceedings of the 6th International Conference on Metaheuristics and Nature Inspired Computing META'16, Marrakech, Morocco, Oct 27-31 2016. p. 368-370.

- Clément Lecat, Corinne Lucet, Chu-Min Li, ”New Lower Bound for the Minimum Sum Coloring” Problem. The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17) : 853-859
- Simon Caillard, Laure Brisoux-Devendeville, Corinne Lucet, ”A planning problem with resource constraints in Health Simulation Center”, The World Congress on Global Optimization (WCGO) 8-10 July 2019, Metz, FRANCE.

#### A.6.4 Conférences nationales

- J. Carlier, C. Lucet, O. Théologou, ”Une méthode de décomposition pour l'évaluation de la fiabilité des réseaux, pour le problème tous terminaux”, Colloque Franco-Suisse de Recherche Opérationnelle, AFCET, Paris, 11-13 Septembre 1991.
- C. Lucet, F. Mendes, A. Moukrim, « Méthode de décomposition appliquée à la coloration de graphes », 4ème Congrès de la ROADEF, Paris, Février 20-22 2002.
- C. Lucet, F. Mendes, A. Moukrim, « Résolution du problème d'allocation de fréquence et coloration de graphes », 5ème Congrès de la ROADEF, Avignon, Février 26-28 2003.
- C. Lucet, F. Mendes, A. Moukrim, « Résolution du problème de coloration de graphes par une méthode exacte de décomposition linéaire », Conférence MAJECSTIC 2004, Calais, 13-15 octobre 2004.
- G. Hardy, C. Lucet, N. Limnios, « Fiabilité des réseaux et Diagrammes de décision binaires », 6ème Congrès de la ROADEF, Tour, Février 06-08 2005.
- F. Mendes, C. Lucet, A. Moukrim « Planification d'horaires de travail dans un centre d'appels : contraintes de charge et gestion de la multi-compétences », 6ème Congrès de la ROADEF, Tour, Février 06-08 2005.
- G. Hardy, C. Lucet, N. Limnios, « Topologie et fiabilité de réseaux via les Diagrammes de Décision Binaires », 7ème Congrès de la ROADEF, Lille, Février 06-08 2006.
- Y. Li, C. Lucet, A. Moukrim, K. Sghiouer, “Somme coloration minimale d'un graphe”, 10ème Congrès de la ROADEF, Nancy, Février 10-12 2009.
- Y. Li, C. Lucet, A. Moukrim, K. Sghiouer, “Bornes inférieures pour la somme coloration de graphe”, 11ème Congrès de la ROADEF, Toulouse, Février 24-26 2010.
- K. Sghiouer, Y. Li, C. Lucet, A. Moukrim, “Algorithme mémétique pour la somme coloration de graphe”, 12ème Congrès de la ROADEF, Saint Etienne, Mars 2011.
- A. Abdoul Soukour, L. Devendeville, C. Lucet, A. Moukrim, “Planification dans le domaine de la sûreté aérienne », 12ème Congrès de la ROADEF, Saint Etienne, Mars 2011.
- Abdoul-Soukour A., Devendeville L., Lucet C., Moukrim A., Un algorithme génétique dédié à la planification dans le domaine de la sûreté aérienne. In ROADEF12, p. 56-57, Angers, France, Avril 2012.
- Lecat C., Li C.M., Lucet C., Li Y., Comparaison de méthodes de résolution pour le problème

de somme coloration, dans les actes des Onzièmes Journées Francophones de Programmation par Contraintes (JFPC-2015), p. 204-213, 2015.

- Lecat C., Li C.M., Lucet C., Li Y., optimisation de la planification de processus en conflit de ressource : résolution par solveur minsat. ROADEF 2015 Conférence, Marseille 23-25 février 2015.
- Clément Lecat, Corinne Lucet and Chu-Min Li : Problème de Somme Coloration Minimale : Borne inférieure de la somme chromatique et borne supérieure de la force du graphe, ROADEF 2017, Metz 22-24 février. (Clément Lecat, Lauréat du Prix Jeune Chercheur)
- Etude de la modélisation en programmation par contraintes pour résoudre le problème de localisation/routage – Laure Brisoux-Devendeville, Corinne Lucet, JFPC 2017, Montreuil-sur-Mer 13-15 juin 2017
- Comparaison de structures protéiques : résolution de la maximisation du recouvrement de cartes de contacts par solveur MaxClique – Olivier Gerard, Corinne Lucet, Chu-Min Li, Montreuil-sur-Mer 13-15 juin 2017
- Considération des motifs dans la détermination de la force d'un graphe – Clément Lecat, Corinne Lucet, Chu-Min Li, JFPC 2017, Montreuil-sur-Mer 13-15 juin 2017
- Résolution du Problème de la Somme Coloration Via une Approche de Type Weighted Partial MaxSAT Clément Lecat, Corinne Lucet et Chu-Min Li, ROADEF 2018, Lorient 21 - 23 février 2018
- An effective variable neighborhood search with perturbation for location-routing problem, Hua Jiang, Corinne Lucet, Laure Devendeville, Chu-Min Li, ROADEF 2019, Le Havre 18-21 Février 2019
- Planification de ressources pour la formation en santé, Simon Caillard, Laure Devendeville, Corinne Lucet, ROADEF 2019, Le Havre 18-21 Février 2019

## A.7 Encadrements et animation recherche

### A.7.1 Encadrement de Thèses

Les cinq thèses que j'ai encadrées entre 1995 et 2017 ont été financées de différentes manières : deux sont des allocations ministérielles, deux sont des allocations du Conseil Régional de Picardie, et une est un contrat CIFRE. Elles ont toutes abouties une soutenance en 40,6 mois en moyenne. Elles ont donné lieu à des collaborations avec les laboratoires HeuDiaSyC et LMA de l'UTC de Compiègne, et avec une entreprise ICTS-France. Elles ont également chacune débouchées sur des publications en conférences et en revue d'audience internationale.

Depuis Novembre 2018, je co-encadre les travaux d'Olivier Gérard sur un problème d'optimisation de la planification du parcours patient dans le milieu hospitalier, avec un financement CIFRE. Il s'agit d'une collaboration avec l'entreprise EVOLUCARE.

- Septembre 1995-Novembre 1998  
Etudiant : Jean-François Manouvrier  
"Méthodes exactes pour des problèmes combinatoires sur les graphes"  
Encadrement : Corinne Lucet, HeuDiaSyC/UTC (100%)  
Soutenue en Novembre 1998.  
Financement : allocation de recherche ministérielle.
- Octobre 2001- Juin 2005  
Etudiante : Florence Mendes  
"Méthodes de décomposition pour des problèmes d'optimisation combinatoire"  
Co-encadrement : Corinne Lucet, LaRIA(MIS)/UPJV (50%) Aziz Moukrim HeuDiaSyC/UTC (50%)  
Financement : allocation de recherche du Conseil Régional de Picardie.
- Novembre 2004 - Novembre 2007  
Etudiant : Gary Hardy  
"Diagrammes de décision binaires pour le calcul de la fiabilité des réseaux de grande taille"  
Co-encadrement : Corinne Lucet, LaRIA/UPJV (50%) Nikolaos Limnios LMAC/UTC (50%)  
Financement : allocation de recherche du Conseil Régional de Picardie.
- Octobre 2009-Novembre 2014  
Etudiant : Anas Abdoul-Soukour  
Co-encadrement : Corinne Lucet, LaRIA/UPJV (33%) Laure Devendeville (MCF, 33%) et Aziz Moukrim (Pr, 33%)  
« PLAN-AIR : Planification des personnels dans la sûreté aérienne »  
Financement : CIFRE. Entreprise ICTS-France
- Octobre 2014-Novembre 2017  
Etudiant : Clément Lecat  
"Réduction de l'espace de recherche du Problème de la Somme Coloration Minimum d'un graphe"  
Encadrement : Corinne Lucet MIS/UPJV (50%), Chumin Li MIS/UPJV (50%)  
Soutenue en Novembre 2017.  
Financement : allocation de recherche ministérielle.
- Novembre 2018  
Etudiant : Olivier Gérard  
"Optimisation du parcours patient en milieu hospitalier"  
Encadrement : Corinne Lucet MIS/UPJV (50%), Gilles Dequen MIS/UPJV (50%)  
Financement : CIFRE. Entreprise Evolucare.

### A.7.2 Encadrements de Stages de Recherche Master

- Etudiant : Delphine Petit (Université de Picardie Jules Verne)  
Encadrement : C. Lucet(50%), A. Moukrim - HeuDiaSyC/UTC(50%)  
Dates : 01/02/2001 - 01/07/2001  
Sujet : Méthode de décomposition pour le problème de coloration de graphe.
- Etudiant : Hamid Bouchebaba (Université de Picardie Jules Verne)  
Encadrement : C. Lucet(50%), Y. Li LARIA/UPJV (50%)  
Dates : 01/02/2002 - 01/09/2002  
Sujet : Recherche de cliques maximales dans un graphe.
- Etudiant : Cyrill Randabell (Université de Picardie Jules Verne)  
Encadrement : C. Lucet(50%), F. Mendes LARIA/UPJV (50%)  
Dates : 01/02/2004 - 01/07/2004  
Sujet : Problème de planification sous contrainte de charge dans un centre d'appels (Coriolis Service).
- Etudiant : Gary Hardy (Université de Picardie Jules Verne)  
Encadrement : C. Lucet  
Dates : 01/02/2004 - 01/07/2004  
Sujet : Diagrammes de décision binaires et fiabilité des réseaux.
- Etudiant : Julie Courtois (Université de Picardie Jules Verne)  
Encadrement : C. Lucet(50%), A. Moukrim - HeuDiaSyC/UTC(50%)  
Dates : 01/02/2006 - 01/07/2006  
Sujet : Heuristiques pour le problème de somme coloration d'un graphe.
- Etudiant : Ting YU (Université de Picardie Jules Verne)  
Encadrement : C. Lucet(50%), Y. Li LARIA/UPJV (50%)  
Dates : 01/03/2010 - 01/08/2010  
Sujet : Recherche Locale pour le problème de la somme coloration de graphe.
- Etudiant : Yassin Chahed Master2 (ISIMA – Clermont Ferrand)  
Encadrement : C. Lucet(33%), Y. Li(33%), C.M. Li(33%)  
Dates : 01/03/2013 - 02/09/2013  
Sujet : Somme coloration de graphe, modélisation et résolution par solveur SAT.
- Etudiant : Sirine Roufaïda Ait Haddadene Master 2 (Université de Lorraine)  
Encadrement : L. Devendeville (50%), C. Lucet (50%)  
Dates : 01/03/2014 - 31/07/2014  
Sujet : Optimisation pour le problème de localisation & routage : modélisation en PLNE et résolution avec le solveur CPLEX.
- Etudiant : Antoine Bette Master1 (Université de Picardie Jules Verne)  
Encadrement : L. Devendeville (50%), C. Lucet(50%)  
Dates : 01/04/2015 - 31/05/2015  
Sujet : Programmation Linéaire en nombres entiers du problème de Localisation Routage(LRP), Résolution CPLEX et comparaison à une métaheuristique développée sur le principe de l'entropie croisée.
- Etudiant : Ines Sbai Master 2 (FSJEG de Jendouba, Tunisie)  
Encadrement : L. Devendeville (50%), C. Lucet(50%)  
Dates : 01/04/2015 - 30/09/2015  
Sujet : Problème de chargement et routage : développement de méta-heuristiques de type VNS

- Etudiant : Mohamed Benjilali Master2 (ENAC Toulouse)  
Encadrement : L. Devendeville (50%), C. Lucet (50%)  
Dates : 15/02/2016- 15/07/2016  
Sujet : Développement d'une métaheuristique de type Cross-Entropy
- Etudiant : Olivier Gérard Master1 (Université de Picardie Jules Verne)  
Encadrement : C. Lucet (50%), C. Li (50%)  
Dates : 15/12/2016- 15/07/2017  
Sujet : Etude de l'alignement de protéines, modélisation par les graphes, recherche de cliques maximales.

### A.7.3 Responsabilités collectives et Organisations

- Co-responsable de l'équipe Graphe et Optimisation Combinatoire du laboratoire MIS depuis Novembre 2018. Animation de l'équipe par l'organisation de journées de séminaires.
- Membre du conseil de laboratoire de 2008 à 2015.
- Co-responsable des séminaires du laboratoire.
- Comité d'organisation de la conférence JFPC et JIAF 2018 à Amiens.
- Comité de programme de Conference on Artificial Intelligence, IJCAI 2018 et 2019
- Animation séance débat sur l'IA - Semaine du Cerveau - Amiens mars 2017 et mars 2018.
- Organisation de la journée "Les projets du laboratoire MIS", de 2010 à 2015.
- Comité d'organisation de la conférence RFIA 2008 à Amiens.
- Journée Industrielle, HEUDIASYC – LaRIA, Compiègne, 30 Novembre 2001. (Organisateur principal Aziz Moukrim, UTC)
- Organisation de la Journée thématique LARIA, ENST-B, IRISA, HEUDIASYC, Amiens le 27 mars 1997.
- Séminaires du groupe PRAO, Compiègne 1996.
- Organisation de la Journée du Groupe Combinatoire de l'AFCEC, HEUDIASYC, Compiègne le 3 Juin 1994.

Relectrice pour les journaux suivants :

- Journal international IEEE Trans. on Reliability
- Journal international Discrete Applied Mathematics
- Journal international Computers & Operations Research
- Journal international Discrete Optimization



## A.8 Valorisation et projets

### A.8.1 Projets de recherche

#### 1 - Pôle Modélisation, Projet 2000.3

Conseil Régional de Picardie (2000/2004)

Responsable pour l'équipe de l'UPJV du projet intitulé « Méthodes de décomposition pour des problèmes d'optimisation combinatoire ». Ce projet fut mené conjointement avec une équipe de chercheurs de l'UTC, dont M. Aziz Moukrim est responsable.

Ce projet a financé une allocation de recherche pour l'UPJV, dont Mme Florence Mendes fut bénéficiaire.

Le montant global du financement hors allocation était de 36 k€

Collaborations Entreprises :

- CORIOLIS Service (centre d'appels à Amiens) pour l'étude de la planification d'emploi du temps sous contraintes de charges)
- La Providence (Institut d'enseignement catholique à Amiens) pour l'étude de la planification d'examens.

#### 2 - Pôle DIVA, Projet 2004.2

Conseil Régional de Picardie (2004/2007)

Responsable pour l'équipe de l'UPJV du projet intitulé « Sûreté de fonctionnement des réseaux de grande taille par les diagrammes de décision binaires » du. Ce projet était une collaboration entre le LaRIA/UPJV, le Laboratoire de Mathématiques Appliquées de l'UTC et du laboratoire HeuDiasyc de l'UTC. M. Nikolaos Limnios est responsable de l'équipe de l'UTC.

Ce projet a financé une allocation de recherche pour l'UPJV, dont M. Gary Hardy fut bénéficiaire.

Le montant global du financement hors allocation était de 39 k €

Collaborations Entreprises : THALES Communications

#### 3 - Projet SCOOP - "Somme COIOration de graphes et aPplications"

Conseil Régional de Picardie (2007/2010)

Responsable pour l'équipe de l'UPJV du projet intitulé . Ce projet était une collaboration entre le laboratoire MIS/UPJV, et le Laboratoire Heudiasyc de l'UTC (M. Aziz Moukrim, UTC, porteur du projet)

Ce projet a financé une allocation de recherche pour l'UTC dont Mme Kaoutar Sghiouer fut bénéficiaire.

Le montant global du financement hors allocation était de 40 k€

#### 4 - PLAN-AIR - "Planification d'emplois du temps dans le domaine de la sécurité aéroportuaire"

Entreprise ICTS-France (2009/2012)

Collaboration entre l'entreprise ICTS-France chargée de la sécurité aéroportuaire (CDG, Orly, Nice, Toulouse) et les laboratoires de recherche MIS/UPJV et HeuDiasyc/UTC.

Ce projet été accompagné d'un financement de thèse en CIFRE, dont a bénéficié M. Anas Abdoul Soukour.

Le montant global du financement hors allocation était de 30 k€

#### 5 - PRIMA - "Problèmes de localisatIon et routage en milieu urbain"

Conseil Régional de Picardie (2011/2014)

Responsable pour l'équipe de l'UPJV du projet intitulé . Ce projet était une collaboration entre le laboratoire MIS/UPJV, et le Laboratoire Heudiasyc de l'UTC (M. Aziz Moukrim, UTC, porteur du projet)

Ce projet a financé une allocation de recherche pour l'UTC dont M. Sohaib Afifi fut bénéficiaire.

Le montant global du financement hors allocation était de 40 k€

**6 - NF3D - "Optimisation d'ordonnancement de tâches dans un process industriel"**

Conseil Régional de Picardie (2011/2014)  
Responsable de la partie optimisation du projet piloté par l'entreprise Pochet de Courval.  
Ce projet finance un postdoc pour le laboratoire MIS.

**7 - SimuStorE - "Plannification pour un centre de pédagogie santé"**

Financé par le centre de pédagogie active SimuSanté - Amiens (2017/2018)  
Collaboration entre le centre de pédagogie SimuSanté et le laboratoire MIS (Corinne Lucet et Laure Brisoux-Devendeville)  
A débouché sur une thèse co-financée par SimuSanté et le Conseil Régional des Hauts-de-France.  
Le montant global du financement hors allocation est de 15 k€

**8 - LORH - "Optimisation du parcours patient"**

Financé par l'entreprise EVOLUCARE  
Collaboration Corinne Lucet, Laure Brisoux-Devendeville et Gilles Dequen  
Est accompagné d'une thèse CIFRE (Olivier Gérard)  
Le montant global du financement hors allocation est de 30 k€

## A.9 Activité pédagogiques

**Mots Clés :** Algorithmique et structures de données, Recherche Opérationnelle, Programmation Linéaire, Théorie des Langages.

Mes activités pédagogiques peuvent être séparées en cinq parties :

- Partie 1. Enseignement en L1 et L2, DUT Informatique (IUT Amiens)
- Partie 2. Enseignement en M1, Master Informatique et Master 2 Recherche (UFR Maths-Info Amiens)
- Partie 3. Enseignement en Formation Ouverte à distance M1 (CNAM) et licence professionnelle L3 (CNAM)
- Partie 4. Encadrement de stages de Master 2
- Partie 5. Encadrement de stages de Master 2

**Partie 1 :** cette première partie est la principale, car mon poste est attaché au département informatique de l'IUT d'Amiens. La totalité de mon service est donc effectué au sein de ce département. Je suis responsable du module AP1 (Algorithmique et Programmation) en première année de DUT Informatique, qui accueille environ 180 étudiants (6 groupes). Le But de cet enseignement est d'initier les étudiants à l'écriture d'algorithmes en leur transmettant lors des cours magistraux (10h CM) l'intérêt et l'aspect crucial de l'algorithmique en informatique, en les faisant écrire sur papier des algorithmes simples utilisant les principaux schémas de contrôle et structures de données élémentaires lors des Travaux Dirigés (20h TD) puis en lors faisant appliquer ces algorithmes à la programmation en langage Pascal lors des séances de Travaux Pratiques (20h TP). Il s'agit souvent d'une première expérience en programmation pour les étudiants de première année et il est vraiment important que des bases solides soient posées à ce stade de leur formation. J'enseigne ce module depuis ma nomination en 1994. Toujours au sein du département informatique, je suis responsable du module « Théorie des langages », dispensé actuellement à l'ensemble des étudiants de deuxième année. Nous faisons dans ce module (6h CM, 20h TD) une introduction à l'analyse lexicale et syntaxique. Nous travaillons sur une définition formelle des langages rationnels, une étude des expression régulière (en voyant la version étendue avec des outils comme grep), et la manipulation des automates à états finis (les principales opérations). J'enseigne ce module sous sa nouvelle forme depuis cette année, mais l'enseignais auparavant depuis trois ans sous forme de module optionnel (6h CM, 20h TD, 10h TP). J'étais également responsable du module de recherche opérationnelle qui fut d'abord dispensé à l'ensemble des étudiants, puis de manière optionnelle, et abandonné avec la nouvelle maquette du DUT. Ce module présentait les algorithmes fondamentaux pour les graphes intervenant dans de nombreuses problématiques (recherche de plus courts chemins, de flots maximaux, etc.)

**Partie 2 :** la seconde partie de mon activité pédagogique de déroule en Master 1ère année d'informatique, dans le module de présentation de la recherche, « graphe et optimisation combinatoire ». Sa durée est mineure (environ 10h) mais permet de faire une présentation de nos travaux de recherche aux étudiants. Dans ce module que je partage avec Mme Yu LI, nous y présentons des approches de résolution d'un problème NP-Difficile comme la coloration de graphe.

**Partie 3 :** la troisième partie est une activité que j'ai abandonnée depuis deux ans, et qui consistait au suivi de la formation à distance d'une dizaine d'auditeurs pour le module RCP101, recherche Opérationnelle et Aide à la Décision, du CNAM Picardie. Ce module a pour objet de présenter des notions de recherche opérationnelle et d'aide à la décision indispensables pour de futurs ingénieurs décideurs, responsables de projets. Cet enseignement fut très enrichissant notamment par la nature des auditeurs qui sont généralement des salariés qui portent un regard professionnel sur les enseignements qui leur sont proposés.

Intitulé	Niveau	CM	TD/étud.	TP/étud.	Total UC	Années
Algorithmique et Programmation	L1	10	20	20	135	2004-2018
Théorie des langages	L2	6	20		89	2007-2011
Recherche Opérationnelle	L2	6	30		39	1998-2007 ; 2014-2018
	L3		30		30	2015-2016
	M2	12			18	2001-2003
Programmation Dynamique	M1	12			18	1998 ; 2000 ; 2005-2006

FIGURE A.4 – Présentation synthétique des enseignements par niveau (L.M.D)

**Partie 4 :** la quatrième activité pédagogique à considérer est l'encadrement de stage de recherche pour des étudiants de master 2. Nous proposons en effet chaque année des sujets de stage de recherche et encadrons un étudiant par an (sauf cette année, pas d'étudiant) pendant une période de 4 à 6 mois.

**Partie 5 :** enfin, j'ai exercé pendant 11 ans la responsabilité des stages en entreprise des D.U.T. 2<sup>de</sup> année, 1999-2011. Cette tâche consistait en :

- Information aux étudiants (env. 100 étudiants)
- Contacts aux entreprises
- Validation des sujets de stages
- Répartition des suivis et organisation des soutenances
- Gestion administrative des conventions

Je suis également en tant que tuteur 6 à 8 stages par an : visite en entreprise, correction de mémoire et jury de soutenance.



## Annexe B

### Sélection d'articles

- [1] Jacques Carlier and Corinne Lucet. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics*, 65(1) :141-156, 1996.
- [2] Corinne Lucet, Jean-Francois Manouvrier, and Jacques Carlier. Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs. *Algorithmica*, 27(3) :316-336, 2000.
- [3] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Trans. Reliability*, 56(3) :506-515, 2007.
- [4] Corinne Lucet, Florence Mendes, and Aziz Moukrim. An exact method for graph coloring. *Computers & OR*, 33 :2189-2207, 2006.
- [5] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, and Yu Li. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics*, 36 :663-670, 2010.
- [6] Abdoul Soukour A., Devendeville L., Lucet C., Moukrim A., A Memetic Algorithm for staff scheduling problem in airport security service. *Expert Syst. Appl*, 40(18) :7504-7512, (2013).
- [7] Clément Lecat, Corinne Lucet, and Chu-Min Li. New lower bound for the minimum sum coloring problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February 4-9, 2017, San Francisco, California, USA., pages 853-859, 2017.
- [8] Clément Lecat, Corinne Lucet, and Chu-Min Li. Minimum sum coloring problem : Upper bounds for the chromatic strength. *Discrete Applied Mathematics*, 233 :71-82, 2017.





ELSEVIER

Discrete Applied Mathematics 65 (1996) 141–156

DISCRETE  
APPLIED  
MATHEMATICS

## A decomposition algorithm for network reliability evaluation<sup>☆</sup>

Jacques Carlier\*, Corinne Lucet

*URA CNRS 817 HEUDIASYC, Université de Technologie de Compiègne, B.P. 649, 60206 Compiègne, France*

Received 9 December 1992; revised 2 July 1993

---

### Abstract

This paper presents an algorithm for computing the  $K$ -terminal reliability of undirected networks, i.e. the probability that a given set of vertices in the network are connected, when edges and nodes fail independently with known probabilities. This algorithm is based on a decomposition method introduced by Rosenthal. It consists in numbering the vertices of the graph so that the successive boundary sets are as small as possible and in evaluating the probabilities of appropriate classes of boundary sets. We show that for the all terminal reliability problem these classes are the partitions of the boundary sets and we describe them for the general problem. Our computational results are so conclusive that networks much larger than those presented before can now be treated.

*Keywords:* Network reliability; Factoring; Reductions; Network decomposition; Boundary set; Algorithm; Computer program

---

### 1. Introduction

The study of network reliability problems started in the 1970s. These problems concern all kinds of networks, such as computer, communication or power networks. When components of the network are subject to random failures, the network may or may not continue functioning after the failures of some components. The probability that the network will function is its reliability. Reliability is an important parameter in both the design and operation of networks [1, 4–6, 10]. We consider here undirected networks, like communication ones. Such a network is traditionally modelled by an undirected graph, whose vertices and edges are subject to statistically independent failures: their components either function or fail with known probabilities. A subset

---

<sup>☆</sup>This work was supported by “le Conseil Régional de Picardie” (Bourse EPR 2934D, Pôle modélisation, projet parallélisme).

\*Corresponding author.



$K$  of vertices that must remain connected is specified. The problem is to compute the probability of such events.

This problem is NP-hard, so enumerative methods are used frequently. Factoring is one such method: it consists in fixing successively the state of particular edges, until the state of the network can be determined. This procedure corresponds to the construction of a binary tree. Factoring is an exponential method, but it becomes very effective by performing reductions. These reductions decrease the size of the network in polynomial time while preserving its reliability. The development of such methods stemmed from the works of Satyanarayana and Wood [6].

Better performances can be achieved by our decomposition method. It consists in numbering the vertices of the graph and in building the corresponding subgraphs and their boundary sets. The boundary vertices can be connected in various ways, according to the states of the subgraph elements.

This connectivity can be summarized by appropriate classes which are, for instance, the partitions of the boundary sets for the all terminal reliability problem. Evaluating the class probabilities makes it possible to compute the network reliability. This can be done more efficiently if the successive boundary sets are the smallest possible. This is confirmed by our numerical experiments which show the superiority of the method over classical ones.

This paper is organized as follows. Section 2 presents basic definitions and notation. Section 3 is a short survey of factoring and reductions methods. In Section 4 we describe the decomposition method for the all terminal reliability problem. The necessary elements for the generalization of the decomposition method are given in Section 5. Section 6 describes briefly an implementation in C.

## 2. Definitions and notation

### Notation

$G$	$(V, E)$ an undirected graph
$V$	the entire set of vertices in $G$
$N$	the number of vertices of $V$
$E$	the entire set of edges in $G$
$p_e$	the reliability of vertex or edge $e$
$q_e$	$1 - p_e$
$K$	a subset of vertices of $V$
$H$	a subgraph of $G$
$F$	the boundary set of $H$ (see Fig. 2)
$[ \dots ] [ \dots ] \dots [ \dots ]$	a partition of $F$
$[xy \dots ]$	a block, that is a subset of $F$
$[xy \dots ]^K$	a block containing at least a vertex of $K$
$R(G_K)$	$K$ -terminal reliability of $G$ : Probability that all $K$ -vertices are connected

The network is modelled by a graph  $G = (V, E)$  and a subset of  $V : K$ . The problem is to compute the  $K$ -terminal reliability of the graph  $G$ , whose vertices and edges fail independently with known probabilities.  $K$  can be every subset of  $V$  such that  $2 \leq |K| \leq |V|$ . If  $K = V$ , it is the all terminal reliability problem.

### 3. Factoring and reductions

Factoring is an enumerative method [10, 3]. Given an undirected graph with perfect vertices, i.e. with vertices which cannot fail, factoring consists in picking an edge and decomposing the original problem with respect to the two possible states of the edge. The network reliability can be written:

$$R(G_K) = p_e \cdot R(G_K^* \cdot e) + (1 - p_e) \cdot R(G_K - e),$$

where  $G_K^* \cdot e$  is  $G$  with edge  $e = (u, v)$  contracted and  $K$  appropriately modified (if  $u \in K$  or  $v \in K$ , then the new vertex  $w = u \cup v$  will be a  $K$ -vertex), and  $G_K - e$  is  $G$  with  $e$  deleted. The reliability of the original network can be computed iteratively, by successive applications of the above formula to the induced graphs, until it is possible to compute directly the reliability of final graphs. This procedure corresponds to the construction of a binary computational tree whose nodes are some graphs; its leaves are graphs with known reliabilities. For certain graphs like series-parallel ones, reliability can be easily and polynomially computed; leaves belong to this category. Factoring is an exponential method but it becomes efficient by performing reductions.

Reductions decrease the size ( $|V| + |E|$ ) of a graph while preserving its reliability. Their application to a graph  $G_K$  results in a reduced graph  $G'_K$ , such that  $R(G_K) = \Omega R(G'_K)$ , where  $\Omega$  is a multiplicative factor dependent on the reductions. In particular, there are simple reductions (parallel, series and of degree-2) and polygon-to-chain reductions introduced by Satyanarayana [6]. Using the simple and polygon-to-chain reductions, series-parallel graphs can be reduced to a simple edge in linear time; their reliability can thus be computed in linear time.

Decomposing a graph into its biconnected (resp. triconnected) components is called biconnected (resp. triconnected) decomposition. Biconnected and triconnected components are computed in linear time using the relevant algorithm of Tarjan [7]. Biconnected decomposition, integrated into factoring and reductions has been implemented by Theologou [8, 9, 2] using an efficient data structure. The author also explains how to treat node failures. The corresponding program makes it possible to compute the reliability of graph with 30 nodes and 50 edges.

### 4. The decomposition method for the all terminal reliability problem

In this section, we state and explain the decomposition method for the all terminal reliability problem. For this problem, all vertices are perfect and terminal. The edges

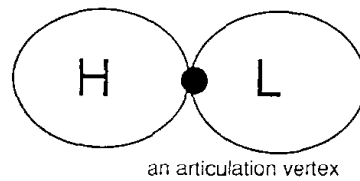


Fig. 1. An articulation vertex.

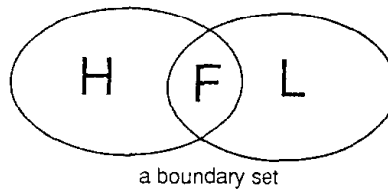


Fig. 2. A boundary set.

are the only elements which can fail. The reliability is the probability that the network is connected. In Section 5, all necessary elements for the general problem are given.

#### 4.1. Subnetwork and boundary set

A subnetwork of  $G = (V, E)$ , is a network  $H = (V', E')$  such that  $V \supseteq V'$ , and  $E' = E \cap (V' \times V')$ . Let  $H$  be any subnetwork of  $G$ , and  $L = (V'', E'')$  the complement graph of  $H$  in  $G$ , i.e.  $V' \cup V'' = V$ ;  $E' \cup E'' = E$  and  $E' \cap E'' = \emptyset$ . Then, the set of vertices  $F = V' \cap V''$ , is called the *boundary set* of  $H$ . In Fig. 1,  $F$  is reduced to only one vertex, when in Fig. 2,  $|F| \geq 2$ .

#### 4.2. Decomposition principle

If  $G$  has an articulation vertex, then  $G$  can be decomposed into two graphs  $H$  and  $L$  such that  $R(G) = R(H) \times R(L)$  (Fig. 1). This decomposition technique was generalized by Rosenthal [4]. He considered a subgraph  $H$  and its complement  $L$  such that  $H$  and  $L$  are separated by the boundary set  $F$ . The author proposes to associate with  $H$  and  $L$  some classes depending on the cardinality of  $F$ . For instance, for the all terminal reliability problem, these classes are the partitions of  $F$ . Then the reliability is computed by associating the classes of  $H$  with the classes of  $L$ .

#### 4.3. Elementary events

The vector  $\mathfrak{G}$  of the element states is an elementary event. Its probability is:  $\Pr(\mathfrak{G}) = \prod p_e \cdot \prod q_{e'}$  where  $e$  are up edges, i.e. edges which function, and  $e'$  are down edges, i.e. edges which fail. An elementary event  $\mathfrak{G}$  of  $G$  is called a working event if its

up elements induce the working of the network, i.e. if the vertices of  $K$  remain in the same connected component. Otherwise, it is a failure event. The network reliability can be written  $R(G_K) = \sum \Pr(\mathfrak{G}_S)$ , where the sum is taken over all the working events  $\mathfrak{G}_S$ .

#### 4.4. Events and partitions

The vector  $\mathfrak{S}$  of the element states of the subnetwork  $H$  is an event. Let us consider the connected components  $O_1, O_2, \dots, O_r$  of the partial graph  $\mathbb{H}$  composed of up elements of  $H$ . If one of these connected components does not meet  $F$ ,  $\mathfrak{S}$  is a failure event because all the elementary events which it contains, are failure events. Otherwise, we associate to  $\mathfrak{S}$  the partition of  $F$  into  $r$  blocks:  $B_1 = O_1 \cap F, B_2 = O_2 \cap F, \dots, B_r = O_r \cap F$ . We define an equivalency between the events of  $H$ : two events are equivalent if their associated partitions are the same. Hence, the classes are the partitions of  $F$ .

#### 4.5. Example

In Fig. 3,  $F = \{u, v, w\}$  and  $\mathbb{H}$  is reduced to the up elements of an event  $\mathfrak{S}$ . The vertices  $u$  and  $v$  are connected via  $\mathbb{H}$  and  $w$  is isolated from the others. So  $u$  and  $v$  are in the same block whenever  $w$  is in its proper block; the corresponding class will be denoted  $[uv][w]$ . Here  $F = \{u, v, w\}$ , so there are five classes:  $[u v w]; [u w][v]; [u][v w]; [u v][w]; [u][v][w]$ . The probability of a class is the sum of the probabilities of the events belonging to it. For the initial graph there is only one class and its associated probability is the network reliability.

#### 4.6. General algorithm

Vertices are numbered from 1 to  $N$  ( $|V| = N$ ). The function  $vertex: [1 \dots N] \rightarrow V$  gives the vertex associated with each number. We denote  $G_k$  the subnetwork induced by vertices  $vertex(1), \dots, vertex(k)$ . There are  $N$  subnetworks  $G_1, G_2, \dots, G_N$  and  $N$  correspondent boundary sets  $F_1, F_2, \dots, F_N$ . Let us denote  $C_{k,s}$  the  $s$ th class of  $G_k$ . To subnetwork  $G_N$  corresponds only one class denoted  $C_{N,1}$ , whose associated

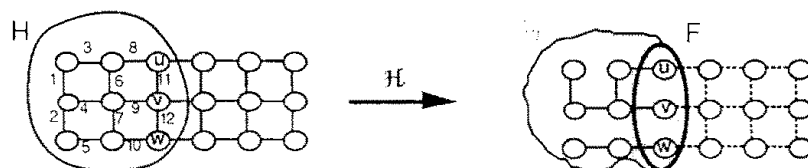


Fig. 3. The event  $\mathfrak{S}$  fixes the states of the twelve edges of  $H$ .  $\mathbb{H}$  is a partial graph of  $H$  reduced to the up edges of the event  $\mathfrak{S} = \langle 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0 \rangle$ .

probability is the network reliability. The method consists in computing iteratively the probabilities of the classes of  $G_2, \dots, G_N$ . Hence,  $\Pr(C_{N,1})$  is reached in  $N - 1$  iterations.

At the  $k$ th iteration, the  $vertex(k)$  and its incident edges are added to  $G_{k-1}$  in order to build  $G_k$  and  $F_k$ . Let us call  $k$ -links the edges incident to  $vertex(k)$  and to some vertices of  $F_{k-1}$  ( $a$  and  $b$  are examples of  $k$ -links in Fig. 4). The classes of  $G_k$  and their associated probabilities are computed from those of  $G_{k-1}$  and of the  $k$ -links. One class of  $G_{k-1}$  induces several classes of  $G_k$  depending on the states of the  $k$ -links. These states induce the connection or the disconnection of  $vertex(k)$  to the different blocks of the class. A block is either connected or not to  $vertex(k)$ . Thus, for a class with  $j$  blocks, the maximum number of connection states to enumerate is  $2^j$ . This maximum number is reached when all blocks of the class have some vertices connected with  $vertex(k)$ .

### General algorithm

```

Init:   Number the vertices of the graph
           $F_1 = \{vertex(1)\}; \Pr(C_{1,1}) = 1;$ 
Begin   for  $k = 2$  to  $N$ 
            Add  $vertex(k)$  to  $G_{k-1}$ 
            Compute  $F_k$ 
             $T(|F_{k-1}|) =$  number of classes of  $G_{k-1}$ 
            for  $s = 1$  to  $T(|F_{k-1}|)$ 
              Generate the class  $C_{k-1,s}$  of  $G_{k-1}$  and compute  $j$ , its
              number of blocks  $\{C_{k-1,s} \rightarrow B_1, B_2, \dots, B_j\}$ 
              for  $m = 1$  to  $2^j$ 
                 $\{m$  describes the connection with  $vertex(k)$  for each
                block  $B_1, B_2, \dots, B_j\}$ 
                Associate to  $B_1, B_2, \dots, B_j$  and  $m$  a class of  $G_k$ .
                Compute its number and its probability
              endfor-m
            endfor-s
          endfor-k
           $R(G) = \Pr(C_{N,1})$ 
end

```

### Example.

In Fig. 4,  $F_{k-1} = \{u, v\}$  and  $F_k = \{u, v, w\}$  such that  $w = vertex(k)$ .  $a$  and  $b$  are the  $k$ -links. The different classes of  $G_{k-1}$  are:

- $C_{k-1,1} = [uv]$  which induces  $[uvw]$  if  $a$  and  $b$  are not both down, and  $[uv][w]$  otherwise. The block  $[uv]$  is connected to  $w$ , then there are  $2^1$  states to enumerate.
- $C_{k-1,2} = [u][v]$  which induces  $[uvw]$  if  $a$  and  $b$  are up,  $[uw][v]$  if  $a$  is up and  $b$  is down,  $[u][vw]$  if  $a$  is down and  $b$  is up, and  $[u][v][w]$  if  $a$  and  $b$  are both down. The

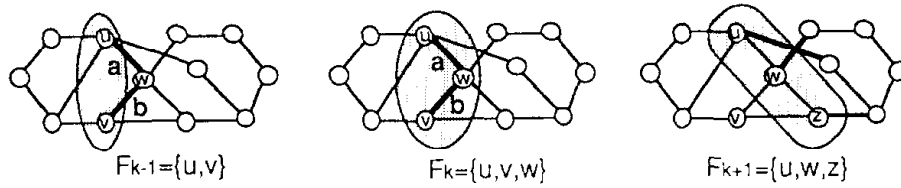


Fig. 4. Partitions of  $F_k$  and their associated probabilities are computed from partitions of  $F_{k-1}$  and the  $k$ -links.

blocks  $[u]$  and  $[v]$  are connected to  $w$  with the two  $k$ -links,  $a$  and  $b$ , then there are  $2^2$  states to enumerate.

The probability of  $C_{k,1} = [u v w]$  is given by the formula:

$$\Pr(C_{k,1}) = \Pr(C_{k-1,1}) \cdot (1 - q_a \cdot q_b) + \Pr(C_{k-1,2}) \cdot p_a \cdot p_b.$$

#### 4.7. Classes numbering

A very important problem in this decomposition method is the classes numbering. All information about a class must be easily and directly accessible. The classes are ordered as follows: at first the classes with a same number  $j$  of blocks are collected in a set  $A(i, j)$ , where  $i$  is the number of vertices of the boundary set  $F_k$ . Next, these sets are ordered in an ascending order of  $j$ . Finally, each  $A(i, j)$  is ordered depending on the order of  $A(i-1, j)$  and  $A(i-1, j-1)$  ( $|A(i, j)| = j \cdot |A(i-1, j)| + |A(i-1, j-1)|$ ). Below we explain this technique using an example. This method makes it possible to have a total order for the classes numbered from 1 to  $\sum_{j=1}^i |A(i, j)|$ .

#### Example.

For  $F_k = \{u\}$ , the class is

$$1 \quad \boxed{[u]}^{A(1,1)}$$

For  $F_k = \{u, v\}$ , the ordering of the classes is

$$1 \quad \boxed{[u v]}^{A(2,1)} \quad v \text{ has been added to the 1st block of } [u]$$

$$2 \quad \boxed{[u] [v]}^{A(2,2)} \quad \text{the block } [v] \text{ has been added to } [u]$$

For  $F_k = \{u, v, w\}$ , the ordering of the classes is

$$1 \quad \boxed{[u v w]}^{A(3,1)} \quad w \text{ has been added to the 1st block of } [u v]$$

2	$[u\ w] [v]$	w has been added to the 1st block of $[u] [v]$
3	$[u] [v\ w]$	w has been added to the 2nd block of $[u] [v]$
4	$[u\ v] [w]$	the block $[w]$ has been added to $[u\ v]$
$A(3, 3)$		
5	$[u] [v] [w]$	the block $[w]$ has been added to $[u] [v]$

We present the following iterative algorithms in  $O(|F_k|)$  to find the class corresponding to a number (see Algorithm 1) and to find the number corresponding to a class (see Algorithm 2). These algorithms are used for enumerating the classes, and for storing the associated probabilities when a new class of  $F_k$  is generated from a class of  $F_{k-1}$ . Therefore, it is essential to have efficient algorithms.

The vertices of  $F_k$  are numbered from 1 to  $|F_k|$ . A class is represented by an array PART[ ]. PART[x] is the block number of the xth vertex of  $F_k$ . For instance, if  $F_k = \{u, v, w\}$ , then the class  $[u\ w] [v]$  is represented by

	1	2	3	4
PART	1	2	1	

The first vertex  $u$  of  $F_k$  is in block 1, the second vertex  $v$  is in block 2 and the third vertex  $w$  is in block 1.

In the following algorithm, let us denote by NUMBER the number of a class,  $j$  the number of blocks of the class NUMBER, and Num the number of the class among the classes with  $j$  blocks. Therefore Num would be defined by the following relation:

$$\text{NUMBER} = \text{Num} + \sum_{h=1}^{j-1} |A(|F_k|, h)|.$$

**Algorithm 1**

**class from number in  $O(|F_k|)$**

*Data:*  $|F_k|$

NUMBER the class number

*Result:* PART[ ], the class

**Begin**

$i := |F_k|$

$j :=$  the number of blocks of the class NUMBER

Num := the number of the class NUMBER among the classes of  $j$  blocks.

**while**  $i < > 1$  **do**

**if**  $\text{Num} \leq |A(i - 1, j)| \cdot j$  **then**

PART[i] := Num - Int((Num - 1)/j) · j

Num := Int((Num - 1)/j) + 1

```

else
    Num:= Num - |A(i - 1,j)|·j
    PART[i]:= j
    j:= j - 1
i:= i - 1
endwhile
PART[1]:= 1
End
    
```

**Algorithm 2**

```

number from class in  $O(|F_k|)$ 
Data: PART[ ], a class of  $F_k$ 
Result: NUMBER the class number
Begin
Num:= 1;
j:= 1
for i:= 2 to  $|F_k|$  do
    if PART[i] = j + 1 then
        j:= j + 1
        Num:= Num + |A(i - 1,j)|·j
    else
        Num:= (Num - 1)·j + PART[i]
    endifor
    {j is the number of blocks of the given class
    PART[ ] and Num the number of the class
    among the classes with j blocks.}
    NUMBER:= Num +  $\sum |A(|F_k|, h)|$  h:= 1 to j - 1
End
    
```

4.8. Complexity

For the *all terminal reliability problem*, the classes of  $G_k$  correspond to the partitions of  $F_k$ . Thereby, their number increases exponentially with the size of the boundary set  $F_k$ .

The *space complexity* depends on the maximum number of classes enumerated during one iteration. It depends also on the maximum size  $F_{\max}$  of the boundary sets  $F_k$ . To a set with  $i$  elements correspond  $T(i)$  partitions:

$$T(i) = \sum_{j=1}^i |A(i, j)|,$$

where  $|A(i, j)|$  is the Stirling number of second kind, given by the recursive formula:

$$|A(i, 1)| = 1 \text{ and for } 2 \leq j \leq i, |A(i, j)| = j \cdot |A(i - 1, j)| + |A(i - 1, j - 1)|.$$



IFI	1	2	3	4	5	6	7	8	9	10	11	12
number of partitions	1	2	5	15	52	203	877	4.140	21.147	115.975	658.586	4.054.653

Fig. 5(a) The total number of partitions according to the size of  $|F_k|$ .

		Number of blocks											
		1	2	3	4	5	6	7	8	9	10	11	12
Cardinality of F	1	1											
	2	1	1										
	3	1	3	1									
	4	1	7	6	1								
	5	1	15	25	10	1							
	6	1	31	90	65	15	1						
	7	1	63	301	350	140	21	1					
	8	1	127	966	1701	1050	266	28	1				
	9	1	255	3025	7770	6951	2646	462	36	1			
	10	1	551	9330	34105	42525	22827	5880	750	45	1		
	11	1	1103	28541	145750	246730	179487	43387	11880	1155	551	1	
	12	1	2207	86726	611541	1379400	1323652	483196	138427	22275	6665	562	1

Fig. 5(b) The number of partitions according to the size of the set  $F_k$  and the number of blocks.

These numbers correspond to the cardinalities of the sets  $A(i, j)$  (see Fig. 5(b)). As a consequence, the space complexity would be  $O(T(F_{\max}))$ .

We use  $OIC(k)$ , to denote the complexity in running time of the  $k$ th iteration.  $OIC(k)$  depends on the number of classes and on the number of connection states (cf. Section 4.6) to enumerate. Then, in the worst case the complexity of the  $k$ th iteration is

$$OIC(k) = |F_k| \sum_{j=1}^{|F_k|} |A(|F_k|, j)| \cdot 2^j.$$

For a graph with  $N$  vertices, there are  $N - 1$  iterations for computing its reliability. The complexity in running time for the all terminal reliability problem with this

decomposition method is

$$O\left(\sum_{k=1}^{N-1} \text{OIC}(k)\right).$$

#### 4.9. Vertices numbering

In order to reduce the number of classes to enumerate, the size of the boundary sets must be the smallest possible [1]. Let  $F_{\max}$  denote the size of the largest boundary set of the graph.  $F_{\max}$  depends on the vertices numbering. Our objective is to obtain an optimal numbering.

We have tried some heuristics and we have concluded that the following one is the best. This heuristic uses a graph exploring method called the breadth-first search. The vertices are numbered from  $N$  down to 1, in the order in which they are reached during the search.

#### The numbering algorithm

##### Init

- iter the number of iterations
- the initial numbering
- $N$  the number of vertices
- the initial  $F_{\max}$

##### Begin

**for**  $j:= 1$  to iter do

Let  $x$  be an uniform random number in  $[1 \dots N]$ .

Let  $v_1$  and  $v_2$  be two adjacent vertices of  $vertex(x)$ .

Modify  $vertex(\ )$  by exchanging  $v_1$  and  $v_2$ .

Apply BREADTH-FIRST SEARCH beginning from  $vertex(N)$ .

Compute the  $\Delta F_{\max}$ .

If  $\Delta F_{\max} < 0$  then

The new numbering is accepted.

**endfor-j**

##### end

An optimal numbering for networks taken in the literature can be obtained by hand. Similarly, for physical reasons, the numbering of planar networks can also be easily done. This makes it possible to evaluate the efficiency of this algorithm. Our heuristic is tested in some graphs, like street networks, given in this paper. As a result, an optimal numbering is taken. Moreover, the computational time does not increase too much with the number of vertices, so the procedure can be used for large networks, but apparently its convergence is very slow. The number of iterations (iter), which was necessary for achieving an optimal numbering, was between 100 (for street

networks  $3 \times 8$ ) and 5000 (for graphs with 30 vertices and 50 edges). Anyway, the interest of our numbering algorithm is essentially to provide a completely automatic method.

## 5. Generalization

### 5.1. The $K$ -terminal problem with perfect vertices

For the  $K$ -terminal reliability problem with perfect vertices, the disconnection of non-terminal vertices does not induce a failed network. In fact, one component of  $G_k$  contains  $K$ -vertices or not. As a component is represented by a block, each block of a partition must have information about this  $K$ -connection.

**Example.** If  $F_k = \{u, v\}$ , there are 6 classes

$[uv]$   $u$  and  $v$  are connected via  $G_k$ , and are not connected with any  $K$ -vertex,

$[uv]^K$   $u$  and  $v$  are connected via  $G_k$ , and connected with some  $K$ -vertices,

$[u][v]$   $u$  and  $v$  are not connected via  $G_k$ , and are not connected with any  $K$ -vertex,

$[u]^K[v]$   $u$  and  $v$  are not connected via  $G_k$ ,  $u$  is connected with some  $K$ -vertices and not  $v$ ,

$[u][v]^K$   $u$  and  $v$  are not connected via  $G_k$ ,  $v$  is connected with some  $K$ -vertices and not  $u$ ,

$[u]^K[v]^K$   $u$  and  $v$  are not connected via  $G_k$ ,  $u$  and  $v$  are connected with some  $K$ -vertices.

To each  $j$ -blocks partition of the boundary set, correspond  $2^j$  classes of  $G_k$ . The number of classes, for a boundary set  $F_k$ , is:  $\sum_{j=1}^{|F_k|} |A(|F_k|, j)| \cdot 2^j$ . As a result, the complexity in running time of the  $k$ th iteration, for the  $K$ -terminal with perfect vertices reliability problem, is:  $OIC(k) = |F_k| \cdot \sum_{j=1}^{|F_k|} |A(|F_k|, j)| \cdot 2^{2 \cdot j}$ .

### 5.2. The $K$ -terminal problem with failed vertices

In the general  $K$ -terminal problem with failed vertices, classes have to describe the state of boundary vertices (for each vertex, two states are possible: up or down), the connecting state of the up boundary vertices and the  $K$ -connection of blocks.

**Example.** If  $F_k = \{u, v\}$ , there are 10 classes. These classes are:

$[u]$   $u$  is up and not connected with any  $K$ -vertex,  $v$  is down,

$[u]^K$   $u$  is up and connected with some  $K$ -vertices,  $v$  is down,

$[v]$   $v$  is up and not connected with any  $K$ -vertex,  $u$  is down,

$[v]^K$   $v$  is up and connected with some  $K$ -vertices,  $u$  is down,

$[uv]$   $u$  and  $v$  are up, connected via  $G_k$ , and not connected with any  $K$ -vertex,

$[uv]^K$   $u$  and  $v$  are up, connected via  $G_k$ , and connected with some  $K$ -vertices,

$[u][v]$   $u$  and  $v$  are up, not connected via  $G_k$  and not connected with any  $K$ -vertex,

$[u]^K[v]$   $u$  is up and connected with some  $K$ -vertices,  $v$  is up and not connected with any  $K$ -vertex.  $u$  and  $v$  are not connected via  $G_k$ ,

$[u][v]^K$   $u$  is up and not connected with any  $K$ -vertex,  $v$  is up and connected with some  $K$ -vertices.  $u$  and  $v$  are not connected via  $G_k$ ,

F	All term.	K-term perfect v	K-term failed v
1	1	2	2
2	2	8	10
3	5	22	46
4	15	94	226
5	52	454	1.214
6	203	2.430	7.106
7	877	14.214	44.958
8	4.140	89.918	305.090
9	21.147	610.182	2.206.398
<b>10</b>	<b>115.975</b>	<b>4.412.958</b>	<b>16.914.146</b>

Fig. 6. The number of partitions to enumerate grows exponentially with the size of F.

$[u]^K[v]^K$   $u$  is up and connected with some  $K$ -vertices,  $v$  is up and connected with some  $K$ -vertices.  $u$  and  $v$  are not connected via  $G_k$ .

We consider a set  $F_{k,x}$  composed with the up vertices of  $F_k$ . According to the states of boundary vertices, there are  $2^{|F_k|} - 1$  different sets  $F_{k,x}$  and exactly  $C_p^{|F_k|}$  sets  $F_{k,x}$  such that  $|F_{k,x}| = p$  ( $C_p^{|F_k|}$  is the binomial number). To each set  $F_{k,x}$  correspond  $\sum_{j=1}^{|F_{k,x}|} |A(|F_{k,x}|, j)| \cdot 2^j$  classes of  $G_k$ . The total number of classes to enumerate is  $\sum_{p=1}^{|F_k|} C_p^{|F_k|} \sum_{j=1}^p |A(p, j)| \cdot 2^j$ . As a result, the complexity in running time of the  $k$ th iteration, for the  $K$ -terminal reliability problem with failed vertices, is:  $OIC(k) = |F_k| \cdot \sum_{p=1}^{|F_k|} C_p^{|F_k|} \sum_{j=1}^p |A(p, j)| \cdot 2^{2 \cdot j}$  (see Fig. 6).

### 5.3. Size of the treated networks

The limit of this decomposition method is essentially due to the quantity of data to store—mainly the classes probabilities of a boundary set—and not to the running time. For the all terminal reliability problem, all the networks with  $F_{max} \leq 12$  can be treated. However, for the  $K$ -terminal reliability problem with imperfect vertices, all the networks with  $F_{max} \leq 10$  can be treated.

## 6. Computational results

At first, the decomposition method (or the boundary method) was implemented for the all terminal reliability problem. The program was written in PASCAL and run on a PC-486. We tested it on some networks chosen in the literature. Smaller running times were obtained, particularly for the greatest networks, as shown in Table 1, by applying the decomposition method than the factoring with reductions method. Then,

Table 1  
All terminal problem, all examples are extracted from [2]

Examples	Vertices	Edges	Reliability	$F_{\max}$	T1(s)	T2(s)
Arpanet	21	32	0.97062	4	6.51	1.82
SODET1	24	38	0.96177	5	40	7.36
SODET2	26	43	0.96907	5	108.71	4.5
T3 × 10	30	47	0.92405	4	26.6	1.70
T4 × 7	28	45	0.93706	3	—	4.50
EDF2	30	52	0.97533	6	1315	12.80
GEN12	140	220	0.91375	12	—	1932

Note: GEN12 is a randomly generated network. T1 is the running time achieved with factoring and reductions on a VAX 8530. T2 is the running time achieved with the boundary method on a PC 486. Vertices and edges have reliability = 0.9.

Table 2  
K-terminal problem, with failed vertices, all examples are extracted from [2]

Examples	Vertices	Edges	Reliability	$F_{\max}$	T1(s)	T2(s)
T3 × 4	12	17	0.708488	3	0.10	0.20
T3 × 5	15	22	0.691933	3	0.47	0.28
T3 × 6	18	27	0.674832	3	2.10	0.33
T3 × 7	21	32	0.657948	3	10.33	0.40
T3 × 8	24	37	0.641439	3	50.14	0.49
CUBE8	16	24	0.76978	4	4.47	0.30
FRA732	11	21	0.79684	4	12.11	0.80

Note: T1 is the running time achieved with factoring and reductions on a VAX 8530. T2 is the running time achieved with the boundary method on a SPARC station. Vertices and edges have reliability = 0.9.

the decomposition method was implemented for the general problem ( $K$ -terminal with imperfect vertices) in C, on a SPARC 2 (30 Mips) and a Tek-Xpress workstation (Table 2). For the street networks, the difference in performance between the decomposition method and the factoring with reductions method is shown in Fig. 7. For graphs with a same fixed  $F_{\max}$ , the running time exponentially grows with the size  $|V| + |E|$  of the graphs when factoring with reductions method is used. On the contrary, for such graphs, the running time linearly grows with the size  $|V| + |E|$ , when the decomposition method is used. In order to estimate the efficiency of the decomposition method, we tested the program on some generated networks with a fixed  $F_{\max}$ . We give in Table 3 the running times for networks with increasing  $F_{\max}$ . These results show how the time exponentially grows with the size of the boundary sets. As a final remark, we note that factoring and reductions can solve only networks with

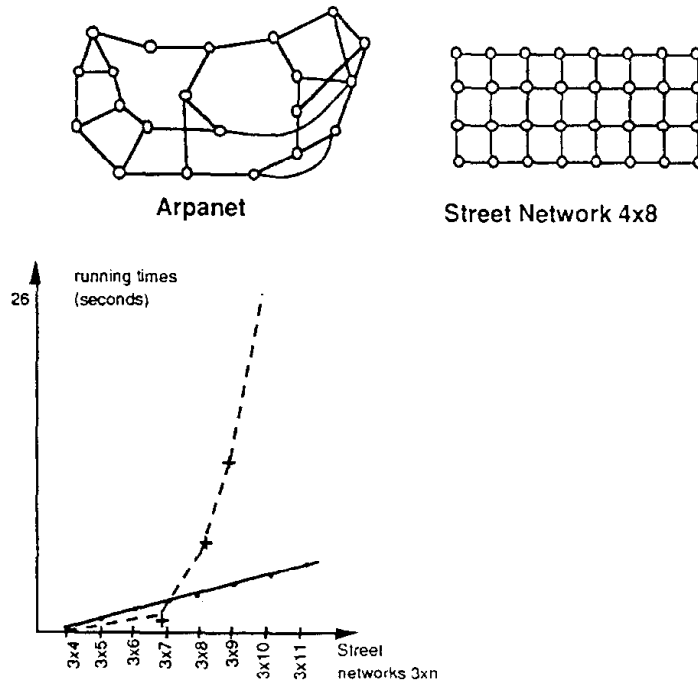


Fig. 7. All terminal reliability problem. Street networks  $3 \times n$ ; (---) running times with the factoring reduction method; (—) running times with the decomposition method.

Table 3  
Running time for the  $K$ -terminal problem with imperfect vertices on a SPARC station (All edges and vertices have reliability = 0.9)

Vertices	Edges	$F_{\max}$	Running time (s)
80	108	5	1.0
90	124	5	2.0
100	155	5	2.3
140	255	5	2.7
80	123	6	5
90	126	6	6
100	134	6	5
140	204	6	6
60	112	7	11
70	137	7	21
80	149	7	30
100	196	7	41
80	150	8	86
90	180	9	182
100	196	10	763

$|F_{\max}| \leq 6$ , when the decomposition method makes it possible to solve the general problem for networks with  $|F_{\max}| \leq 10$ . These results prove the extreme efficiency of this method and the considerable overall improvement in performance on this class of networks problems.

## References

- [1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey, *BIT* 25 (1985) 2–23.
- [2] J.G. Carlier and O.R. Theologou, An efficient program for computing undirected network reliability, in: R. Avenhaus, H. Karkar and M. Rudnianski, eds., *Defense Decision Making* (Springer, Berlin, 1991) 3–13.
- [3] L. Page and J. Perry, A practical implementation of the factoring theorem for network reliability, *IEEE Trans. Reliability* 37 (1988) 259–267.
- [4] A. Rosenthal, Computing the reliability of complex networks, *SIAM J. Appl. Math* 32 (1977) 384–393.
- [5] A. Satyanarayana and M. Chang, Network reliability and the factoring theorem, *Networks*, 1 (1983) 107–120.
- [6] A. Satyanarayana and R.K. Wood, A linear-time algorithm for computing  $K$ -terminal reliability in series parallel networks, *SIAM J. Comput.* 14 (1985) 818–832.
- [7] R. Tarjan, Dividing a graph into triconnected components, *SIAM J. Comput.* 2 (1973) 135–158.
- [8] O.R. Theologou, Contribution to network reliability evaluation, Ph. D Thesis, Department of Computer Science, Université de Technologie de Compiègne (1990) (in French).
- [9] O.R. Theologou and J.G. Carlier, Factoring and reductions for networks with imperfect vertices, *IEEE Trans. Reliability* 40 (1991) 210–217.
- [10] R.K. Wood, Triconnected decomposition for computing  $K$ -terminal network reliability, *Networks* 19 (1989) 203–220.

# Evaluating Network Reliability and 2-Edge-Connected Reliability in Linear Time for Bounded Pathwidth Graphs

C. Lucet,<sup>1</sup> J.-F. Manouvrier,<sup>1</sup> and J. Carlier<sup>1</sup>

**Abstract.** This paper presents a decomposition method for computing the 2-edge-connected reliability of undirected networks. This reliability is defined as the probability that all the vertices of a given graph  $G$  are 2-edge-connected, when edges fail independently with known probabilities. The principle of this method was introduced by Rosenthal in 1977 [1]. For the all terminal reliability problem it consists in enumerating specific state classes of some subnetworks. These classes are represented by the partitions of the boundary sets. For the 2-edge-connected reliability problem these classes are represented by labeled forests whose nodes are the partition blocks and some “unidentified” blocks. Our implementation uses a vertex linear ordering. The computational complexity depends on the number of classes, which depends on the vertex separation number of a given vertex linear ordering. Our computational results show the efficiency of this method when the vertex separation number is smaller than 7.

**Key Words.** Network reliability, 2-Edge-connectivity, Network decomposition, Boundary set, Pathwidth.

**1. Introduction.** There is much literature devoted to the problem of all terminal reliability [2]–[5]. When considering this problem, it is supposed that the network is modeled by an undirected graph  $G = (V, E)$ . Vertices are perfect, but edges can fail with known probabilities. Computing the all terminal reliability means computing the probability that the network remains connected when failures are statistically independent. The aim of this paper is to revisit a decomposition method for the problem of all terminal reliability and to adapt it to 2-edge-connected reliability.

For general networks, all terminal reliability computation is NP-hard [6]. Satyanarayana and Chang [7] and Wood [8] have shown that the factoring algorithm using reductions is more efficient than the classical path or cut enumeration methods for solving it. This is confirmed by the experimental works of Theologou and Carlier [9], but its running time remains prohibitive for large networks. In 1977 Rosenthal presented a decomposition method which is a generalized reduction [1]. It consists in splitting up the network according to a boundary set of vertices and evaluating the probabilities of some subnetwork classes such that the reliability can be achieved by combining some of them. Carlier and Lucet have worked out and tested this method for  $K$ -terminal reliability computation with imperfect edges and vertices [10]. Their results show that it is more efficient than factoring using reductions and that it can be applied to real-world-size networks. In this paper we are concerned with another fundamental measure on networks:

---

<sup>1</sup> UMR CNRS 6599 HEUDIASYC, Université de Technologie de Compiègne, B.P. 20529, 60205 Compiègne cedex, France. {Corinne.Lucet, JF.Manouvrier, Jacques.Carlier}@hds.utc.fr.



2-edge-connected reliability. This is defined as the probability that any two vertices are connected by at least two edge-disjoint paths. Implementation of the decomposition method [11] has shown that it is very efficient for computing standard  $K$ -terminal reliability, even when the vertices of  $G$  are subject to failures. We therefore propose that this decomposition method be used for 2-edge-connected reliability evaluation.

This paper is organized as follows. First, in Section 2, we describe the problem and give some definitions. Then, in Section 3, we present the decomposition method for all terminal reliability. Next, in Section 4, we show that it can be used to compute 2-edge-connected reliability by defining some appropriate subnetwork classes. Finally, in Section 5, we examine the computational complexity of the algorithm using enumeration of these classes and show that it can compute all terminal reliability and 2-edge-connected reliability in linear time for graphs with bounded pathwidth.

## 2. Definitions and Notation

**2.1. Notation.**  $\mathbf{G} = (V, E)$  is an undirected graph, with  $\mathbf{V}$  its set of vertices, and  $\mathbf{E} \subset V \times V$  its set of edges.  $(\mathbf{u}, \mathbf{v})$  denotes an edge of  $E$ , a nonordered pair with  $u \in V$  and  $v \in V$ . Consequently,  $(u, v)$  and  $(v, u)$  denote the same edge.  $\mathbf{p}_e$  is the reliability of the edge  $e$ , and  $\mathbf{q}_e = 1 - p_e$ .  $\mathbf{H}$  and  $\mathbf{L}$  are subgraphs of  $G$ .  $\mathbf{H} \cup \mathbf{L}$  is a graph, i.e., its vertices are the vertices of  $H$  and the vertices of  $L$ , and its edges are the edges of  $H$  and the edges of  $L$ .  $\mathbf{H} \cap \mathbf{L}$  is a graph, i.e., its vertices are the vertices belonging to both  $H$  and  $L$  and its edges are the edges belonging to both  $H$  and  $L$ .  $\mathbf{F}$  is the boundary set of  $H$ .  $[\dots][\dots] \dots [\dots]$  denotes a partition of  $F$ .  $\mathcal{R}(G)$  is the all terminal reliability of  $G$  and  $\mathcal{R}_{2ec}(G)$  is the all terminal 2-edge-connected reliability of  $G$ .  $\mathcal{G}_i$  is a state of  $G$  and  $\mathbf{G}(\mathcal{G}_i)$  its associated partial graph.  $|S|$  is the cardinal of any set  $S$ .

**2.2. Definitions.** A given graph  $G$  is connected if there exists at least one path between any two vertices. It is 2-edge-connected if there exist at least two paths without common edges between any two vertices. Our network model is an undirected stochastic graph  $G = (V, E)$ . Each edge of  $E$  can fail, statistically independently with known probability, but vertices of  $V$  are perfectly reliable. The failure probability of the edge  $e \in E$  is  $q_e$  ( $q_e \in [0, 1]$ ) and its reliability is  $p_e = 1 - q_e$ . A subgraph of a given graph  $G = (V, E)$  is a graph  $G' = (V', E')$  such that  $V' \subset V$  and  $E' = (V' \times V') \cap E$ . A partial graph of a given graph  $G = (V, E)$  is a graph  $G'' = (V, E'')$  such that  $E'' \subset E$ .

Each edge of the stochastic graph is subject to failure. So, as there are two states for an edge (each edge functions or fails), there are  $2^{|E|}$  possible states for the graph. One state  $\mathcal{G}_i$  of the stochastic graph  $G = (V, E)$  is denoted  $\langle s_1, s_2, \dots, s_{|E|} \rangle$  where  $s_e$  stands for the state of edge  $e$ , i.e.,  $s_e = 0$  when edge  $e$  fails and  $s_e = 1$  when it functions. The associated probability of  $\mathcal{G}_i$  is

$$(1) \quad \text{Prob}(\mathcal{G}_i) = \prod_{e \in E} [s_e \cdot p_e + (1 - s_e) \cdot q_e].$$

With each state  $\mathcal{G}_i$  of  $G = (V, E)$  is associated a partial graph  $G(\mathcal{G}_i) = (V, E'')$ , such that  $e \in E''$  if and only if  $e \in E$  and  $s_e = 1$ . In the following we also consider states and partial graphs of subgraphs  $H$  and  $L$  of  $G$ .  $\mathcal{H}_i = \langle s'_1, s'_2, \dots, s'_{|E'|} \rangle$  denotes one state of

$H = (V', E')$  and  $H(\mathcal{H}_i)$  its associated partial graph. The reliability of  $G = (V, E)$  is the probability that  $G$  supports a given operation. For the all terminal reliability problem, this operation requires that any two vertices are able to communicate via at least one operational path:

$$(2) \quad \mathcal{R}(G) = \sum_{G(\mathcal{G}_i) \text{ is connected}} \text{Prob}(\mathcal{G}_i).$$

For the all terminal 2-edge-connected reliability problem, it is necessary that any two vertices of  $V$  are able to communicate via at least two operational paths with no edges in common. So the 2-edge-connected reliability problem consists in evaluating the probability that a given graph is 2-edge-connected when its edges fail independently. This probability can be obtained by summing all the associated probabilities of states  $\mathcal{G}_i$ , such that  $G(\mathcal{G}_i)$  is 2-edge-connected:

$$(3) \quad \mathcal{R}_{2ec}(G) = \sum_{G(\mathcal{G}_i) \text{ is 2-edge-connected}} \text{Prob}(\mathcal{G}_i).$$

### 3. Decomposition Principle for All Terminal Reliability

3.1. *Introduction.* Let  $v \in V$  such that its removal leaves  $G$  disconnected, i.e.,  $v$  is an articulation point of  $G$ . Therefore,  $G$  can be decomposed into two subgraphs  $H$  and  $L$ , such that the vertex set of  $H \cap L$  is  $\{v\}$ ,  $H \cup L = G$  (Figure 1), and  $\mathcal{R}(G) = \mathcal{R}(H) \cdot \mathcal{R}(L)$ .

This technique was extended by Rosenthal in 1977 [1] for the all terminal reliability problem, when the vertex set of  $H \cap L$  is  $F$ , a separator set of the graph, with  $|F| \geq 2$ .  $F$  is the boundary set of  $H$  and  $L$ , and its vertices are called the boundary vertices. It is also supposed that  $H$  and  $L$  have no edges in common.

Rosenthal proposed to associate with  $H$  and  $L$  some state classes depending on  $F$ . One class of  $H$  regroups some of its states according to an equivalence relation. For the all terminal problem, these classes can be modeled by the partitions of  $F$ . Hence, the total information about  $H$  is reduced to these boundary vertex connections.

3.2. *Operating and Failure States of a Subgraph.* Let  $\mathcal{H}_i$  be a state of  $H$  and let  $H(\mathcal{H}_i)$  be the partial subgraph of  $H$  composed of the surviving edges. Let  $F$ -clique be

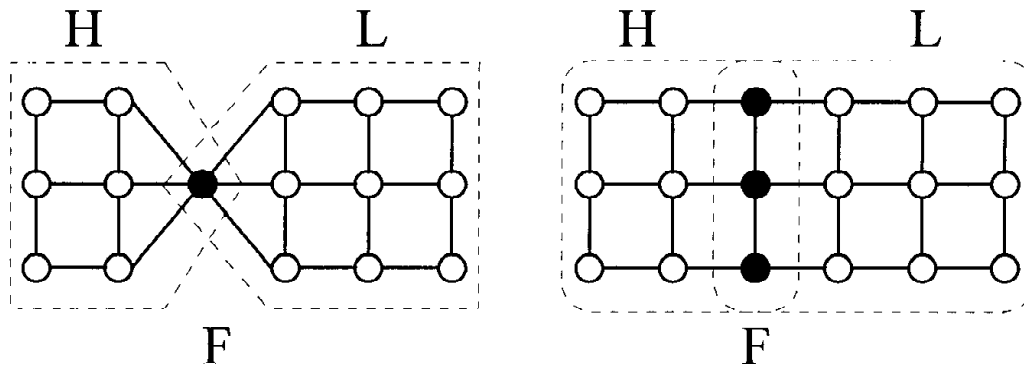
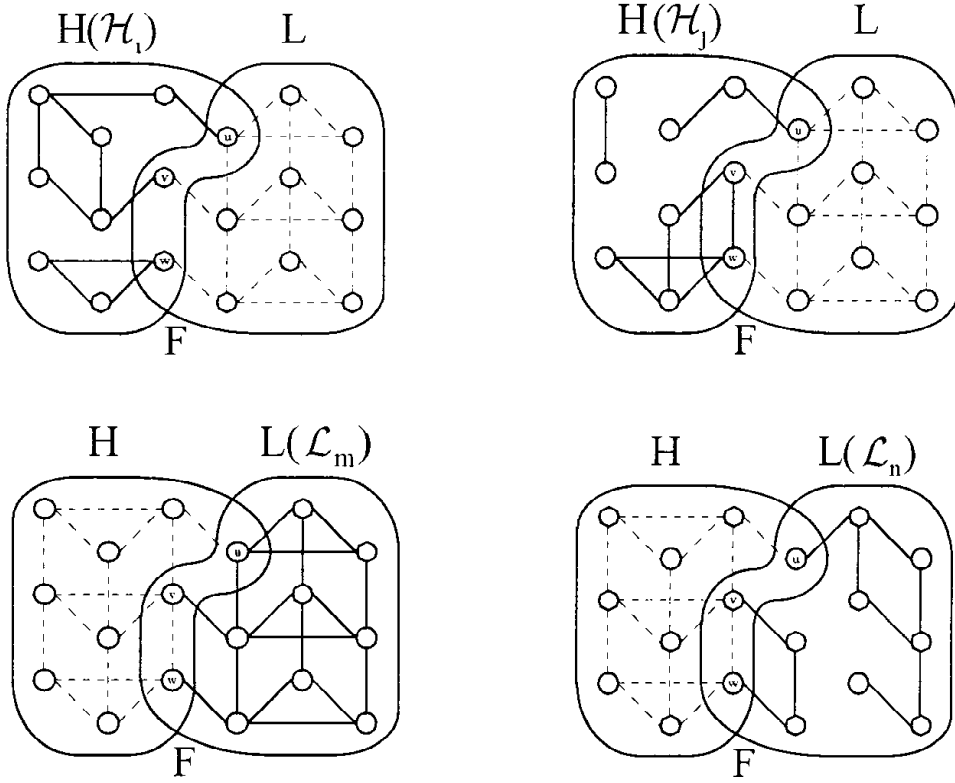


Fig. 1. Decomposition principle.  $|F| = 1$  (articulation point) and  $|F| = 3$ .



**Fig. 2.** State examples for all terminal reliability. Only the operating edges of a state are represented.

the completely connected subgraph  $(F, F \times F)$ . By definition, a state  $\mathcal{H}_i$  of  $H$  is an operating state if  $H(\mathcal{H}_i) \cup (F\text{-clique})$  is a connected graph; that is to say that a state  $\mathcal{L}_j$  of  $L$  could exist such that  $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$  is connected, i.e.,  $\mathcal{H}_i \cup \mathcal{L}_j$  is an operating state  $\mathcal{G}_k$  of  $G$ . Otherwise, if  $H(\mathcal{H}_i) \cup (F\text{-clique})$  is not a connected graph, some of the vertices of  $H(\mathcal{H}_i)$  are permanently disconnected, whatever the topology of  $L$ . In this case  $\mathcal{H}_i$  is called a failure state.

In Figure 2  $\mathcal{H}_i$  is an operating state of  $H$ , whereas  $\mathcal{H}_j$  is a failure state. In this example, the boundary set between  $H$  and  $L$  is  $F = \{u, v, w\}$ :

- $H(\mathcal{H}_j)$  consists of three connected components denoted as  $\mathcal{O}_{H_j,1}$ ,  $\mathcal{O}_{H_j,2}$ , and  $\mathcal{O}_{H_j,3}$ . One of these connected components is disconnected from the boundary set:  $\mathcal{O}_{H_j,1} \cap F = \{u\}$ ,  $\mathcal{O}_{H_j,2} \cap F = \{v, w\}$ ,  $\mathcal{O}_{H_j,3} \cap F = \emptyset$ . Therefore there is no state of  $L$  that allows the vertices of  $\mathcal{O}_{H_j,3}$  to be connected to the others, not even the state  $\mathcal{L}_m$ , which is the state such that all edges of  $L$  function, i.e.,  $\mathcal{H}_j \cup \mathcal{L}_m$  is a failure state of  $G$ . So  $\mathcal{H}_j$  is a failure state of  $H$ .
- $H(\mathcal{H}_i)$  contains two connected components,  $\mathcal{O}_{H_i,1}$  and  $\mathcal{O}_{H_i,2}$ , connected with  $F$ :  $\mathcal{O}_{H_i,1} \cap F = \{u, v\}$ ,  $\mathcal{O}_{H_i,2} \cap F = \{w\}$ . For an operating state such as  $\mathcal{H}_i$ , the network has a possibility of being connected and this possibility depends on the state of the subgraph  $L$ . If the state of the subgraph  $L$  is a failure state, the network cannot be connected. We now consider one operating state of  $L$ .  $L(\mathcal{L}_n)$  is composed of two connected components  $\mathcal{O}_{L_n,1}$  and  $\mathcal{O}_{L_n,2}$ :  $\mathcal{O}_{L_n,1} \cap F = \{u\}$ ,  $\mathcal{O}_{L_n,2} \cap F = \{v, w\}$ .  $H(\mathcal{H}_i) \cup L(\mathcal{L}_n)$  is connected and therefore  $\mathcal{H}_i \cup \mathcal{L}_n$  is an operating state of  $G$ .

The formula for the reliability (2) of our graph can be written:

$$(4) \quad \mathcal{R}(G) = \sum_{\mathcal{H}_i, \mathcal{L}_j / H(\mathcal{H}_i) \cup L(\mathcal{L}_j) \text{ is connected}} \text{Prob}(\mathcal{H}_i) \cdot \text{Prob}(\mathcal{L}_j).$$

PROPOSITION 3.1. *The two following statements are equivalent:*

Statement 1.  $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$  is connected.

Statement 2.  $\mathcal{H}_i$  and  $\mathcal{L}_j$  are both operating states

and

$(F \cap \mathcal{O}_{H_i,1})\text{-clique} \cup \dots \cup (F \cap \mathcal{O}_{H_i,s})\text{-clique} \cup (F \cap \mathcal{O}_{L_j,1})\text{-clique} \cup \dots \cup (F \cap \mathcal{O}_{L_j,t})\text{-clique}$  is connected.

Proposition 3.1 entails that the information required for providing the functioning state of  $G$  is, for two operating states, the manner of connection of the boundary vertices, i.e., their connections via  $H$  and via  $L$ .

3.3. *Class Definition and Equivalence Relation.* We have seen above that the functioning of  $G$  depends on the manner of connection of the boundary vertices, via  $H$  and via  $L$ . Now, certain operating states of a subgraph give the same connectivity for the boundary vertices via  $H$ . Such states are called equivalent states and are grouped in the same class. In Figure 3 the two operating states  $\mathcal{H}_i$  and  $\mathcal{H}_k$  are equivalent states of the subgraph  $H$ , because they provide an identical connectivity of the boundary vertices.

Rosenthal has grouped all the failure states of  $H$  into a failure class, denoted  $\text{DEF}(H)$ , and all the operating states  $\mathcal{H}_i$  of  $H$  into operating classes, according to the manner of connection of the boundary vertices, via the partial subgraph  $H(\mathcal{H}_i)$ . In the case of all terminal reliability, these operating classes are the partitions of  $F$ . One partition is made of several blocks. Each block stands for the intersection between one connected component of  $H(\mathcal{H}_i)$  and  $F$ , i.e., one block contains vertices of  $F$  that belong to the same connected component.

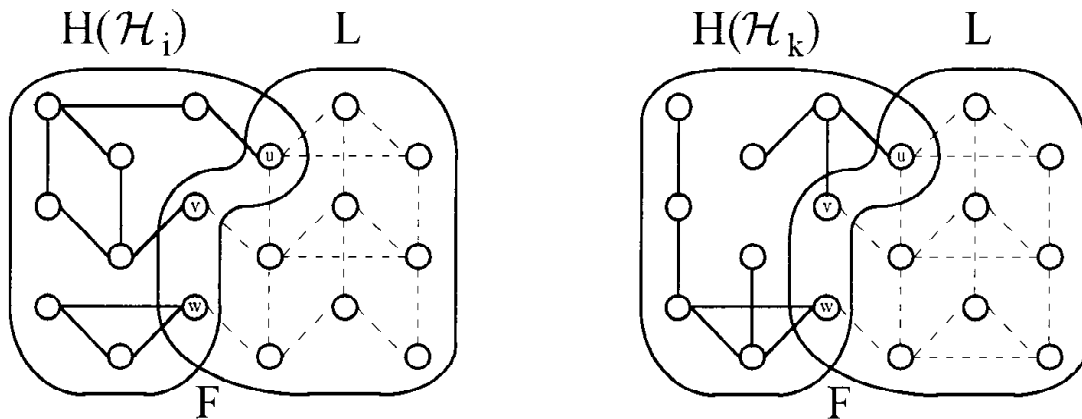


Fig. 3. Two equivalent states for all terminal reliability.  $\mathcal{H}_i$  and  $\mathcal{H}_k$  are two operating states of  $H$  that belong to the same class:  $[uv][w]$ .

We denote  $C_{H,k}$  as the  $k$ th class of  $H$ . For instance, the classes of  $H$  for a boundary set of three vertices  $u$ ,  $v$ , and  $w$  can be the following:

$$\begin{aligned} C_{H,1} &= [uvw], & u, v, \text{ and } w \text{ are connected via } H. \\ C_{H,2} &= [uv][w], & u \text{ and } v \text{ are both connected via } H, \text{ and } w \text{ is disconnected.} \\ C_{H,3} &= [uw][v], & u \text{ and } w \text{ are both connected via } H, \text{ and } v \text{ is disconnected.} \\ C_{H,4} &= [u][vw], & v \text{ and } w \text{ are both connected via } H, \text{ and } u \text{ is disconnected.} \\ C_{H,5} &= [u][v][w], & u, v, \text{ and } w \text{ are disconnected via } H. \end{aligned}$$

The example in Figure 3 shows two equivalent states that belong to the class  $[uv][w]$ . The class  $C_{H,2}$  is a factorization of all states  $\mathcal{H}_i$  composed of two connected components  $\mathcal{O}_{H_i,1}$  and  $\mathcal{O}_{H_i,2}$  with  $\mathcal{O}_{H_i,1} \cap F = \{u, v\}$  and  $\mathcal{O}_{H_i,2} \cap F = \{w\}$ .

**3.4. The Decomposition Principle.** The decomposition algorithm consists in enumerating the operating classes of  $H$  and  $L$  (omitting the failure classes  $\text{DEF}(H)$  and  $\text{DEF}(L)$ ), and in computing their associated probabilities. The associated probability of the class  $C_{H,k}$  is

$$(5) \quad \text{Prob}(C_{H,k}) = \sum_{\mathcal{H}_i / \mathcal{H}_i \in C_{H,k}} \text{Prob}(\mathcal{H}_i).$$

The reliability of  $G$  is computed by combining the compatible classes of  $H$  and  $L$ . Two classes  $C_{H,x}$  and  $C_{L,y}$  are compatible if the connectivity of the boundary set given by  $C_{H,x}$  and the connectivity of the boundary set given by  $C_{L,y}$  provide the connectivity of the whole graph  $G$ .

We report here some of the possible combinations between the classes of two subgraphs  $H$  and  $L$ , separated by a boundary set of three vertices, and we specify the compatible classes. The set of possible classes is the set of partitions, that is  $\{[uvw], [uv][w], [uw][v], [u][vw], [u][v][w]\}$ .

$$\begin{aligned} C_{H,1} &= [uvw] & \text{and} & & C_{L,5} &= [u][v][w] & \text{are compatible classes.} \\ C_{H,2} &= [uv][w] & \text{and} & & C_{L,5} &= [u][v][w] & \text{are not compatible classes.} \\ C_{H,3} &= [uw][v] & \text{and} & & C_{L,3} &= [uw][v] & \text{are not compatible classes.} \\ C_{H,3} &= [uw][v] & \text{and} & & C_{L,4} &= [u][vw] & \text{are compatible classes.} \end{aligned}$$

So there is a factorization of (4) that becomes

$$(6) \quad \mathcal{R}(G) = \sum_{C_{H,x}, C_{L,y} / C_{H,x} \text{ and } C_{L,y} \text{ are compatible}} \text{Prob}(C_{H,x}) \cdot \text{Prob}(C_{L,y}),$$

which is in fact

$$\mathcal{R}(G) = \sum_{C_{H,x}, C_{L,y}, \text{ compatible}} \left[ \sum_{\mathcal{H}_i \in C_{H,x}} \text{Prob}(\mathcal{H}_i) \cdot \sum_{\mathcal{L}_j \in C_{L,y}} \text{Prob}(\mathcal{L}_j) \right].$$

Formula (6) is more efficient than (4), because it reduces the number of multiplications.

**Table 1.** The total number of partitions according to the size of  $F$ .

$ F $	1	2	3	4	5	6	7	8	9	10
Nb_classes	1	2	5	15	52	203	877	4140	21,147	115,975

Rosenthal’s algorithm [1] uses the recurrence formula:

$$(7) \quad \text{Prob}(C_{H3,z}) = \sum_{\substack{C_{H1,x}, C_{H2,y}/C_{H1,x} \text{ and } C_{H2,y} \\ \text{provide the connectivity of } C_{H3,z}}} \text{Prob}(C_{H1,x}) \cdot \text{Prob}(C_{H2,y})$$

with  $H3 = H1 \cup H2$

**Algorithm [1]**

By definition, a subgraph is resolved if the probabilities of all its classes have been computed. Repeat 1 and 2 until  $H3 = G$ .

1. Choose two resolved subgraphs denoted as  $H1$  and  $H2$  such that  $H3 = H1 \cup H2$ .
2. Use (7) to resolve the subgraph  $H3$ .

3.5. *Number of Classes.* The subgraphs  $H$  and  $L$  have an equal number of classes, which is the number of partitions of  $F$ . We denote the Stirling number of the second kind by  $A_{i,j}$ , which is the number of partitions with  $j$  blocks for a set of  $i$  elements. This number grows exponentially with  $i$ , consequently the number of classes grows exponentially with the size of the boundary set  $F$  (see Table 1). We have the recurrent formulae:

$$A_{i,j} = 1 \quad \text{if } j = 1, \quad A_{i,j} = 0 \quad \text{if } 0 < i < j,$$

$$A_{i,j} = j \cdot A_{i-1,j} + A_{i-1,j-1} \quad \text{if } 1 < j \leq i.$$

$$\text{Nb\_classes} = \sum_{j=1}^{|F|} A_{|F|,j}.$$

The 2-edge-connected reliability can be computed in a similar way, but in this case the classes are not simply partitions of the boundary set  $F$ . We describe these classes in the following section.

**4. Classes for All Terminal 2-Edge-Connected Reliability**

4.1. *Introduction.* Our aim now is to use the same decomposition principle for 2-edge-connected (2ec) reliability. Decomposition with an articulation point gives a similar formula to that in Section 3.1:  $\mathcal{R}_{2ec}(G) = \mathcal{R}_{2ec}(H) \cdot \mathcal{R}_{2ec}(L)$ . In this section we extend this principle for a set  $F$  of boundary vertices between two subgraphs  $H$  and  $L$  ( $H \cup L = G$ ) with  $|F| \geq 2$ .  $H$  and  $L$  are assumed to have no edges in common (Figure 1).

The main difficulty here is to define appropriate classes to stand for the sets of equivalent states of a subgraph. Previously, the information contained in a class for reliability computation was the connectivity of the boundary vertices via  $H$ . Now the information required to specify a class for 2ec reliability computation will be the connectivity and the 2-edge-connectivity of the boundary vertices via  $H$ .

4.2. *Operating and Failure States of a Subgraph.* To define operating and failure states, we introduce the notion of a 2-edge-connected-clique. By definition, a subgraph  $G' = (V', E')$  of a graph  $G = (V, E)$  is a 2-edge-connected-clique (2ec-clique) if and only if  $|V'| \neq 2$  and  $G'$  is a clique, or  $|V'| = 2$  and  $G'$  is the multigraph  $G' = (\{u, v\}, \{(u, v), (u, v)\})$ . If a subgraph  $G' = (V', E')$  of a graph  $G = (V, E)$  is a 2ec-clique, then there exist two edge-disjoint paths between any two vertices of  $V'$ .

With regard to 2ec reliability, a state  $\mathcal{H}_i$  is by definition an operating state if  $H(\mathcal{H}_i) \cup (F\text{-}2\text{ec-clique})$  is 2-edge-connected. So a failure state  $\mathcal{H}_j$  is a state which does not permit the whole graph to be 2-edge-connected, whatever the state of  $L$ . Figure 4 summarizes the conditions for operating and nonoperating states for the two problems under discussion.

We again consider the examples of Figure 2. The state  $\mathcal{L}_n$  is a failure state for 2-edge-connectivity although it is an operating state for the connectivity. As we have shown in Section 3.2, a state  $\mathcal{G}_k$  of  $G$  is composed of two states in the two subgraphs  $H$  and

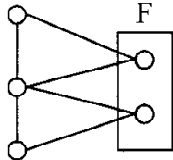
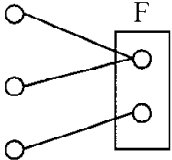
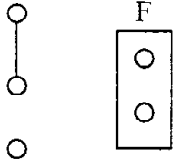
Reliability :		
<p style="text-align: center;"><u>Operating states</u></p> <p style="text-align: center;"><math>H(\mathcal{H}_i) \cup (F\text{-clique})</math> is connected</p> <p style="text-align: center;">=</p> <p style="text-align: center;"><math>\forall v \in (H(\mathcal{H}_i) - F), \exists</math> a path from <math>v</math> to <math>F</math>.</p>	<p style="text-align: center;"><u>Failure states</u></p> <p style="text-align: center;"><math>H(\mathcal{H}_i) \cup (F\text{-clique})</math> is disconnected</p> <p style="text-align: center;">=</p> <p style="text-align: center;"><math>\exists v \in (H(\mathcal{H}_i) - F),</math> <math>\nexists</math> a path from <math>v</math> to <math>F</math>.</p>	
2-edge-connected reliability :		
<p style="text-align: center;"><u>Operating states</u></p> <p style="text-align: center;"><math>H(\mathcal{H}_i) \cup (F\text{-}2\text{ec-clique})</math> is 2-edge-connected</p> <p style="text-align: center;">=</p> <p style="text-align: center;"><math>\forall v \in (H(\mathcal{H}_i) - F), \exists</math> two edge-disjoint paths from <math>v</math> to <math>F</math>.</p>	<p style="text-align: center;"><u>Failure states</u></p> <p style="text-align: center;"><math>H(\mathcal{H}_i) \cup (F\text{-}2\text{ec-clique})</math> is not 2-edge-connected</p> <p style="text-align: center;">=</p> <p style="text-align: center;"><math>\exists v \in (H(\mathcal{H}_i) - F),</math> <math>\nexists</math> two edge-disjoint paths from <math>v</math> to <math>F</math>.</p>	
<p><math>\mathcal{H}_a</math></p> 	<p><math>\mathcal{H}_b</math></p> 	<p><math>\mathcal{H}_c</math></p> 

Fig. 4. Operating and failure states.

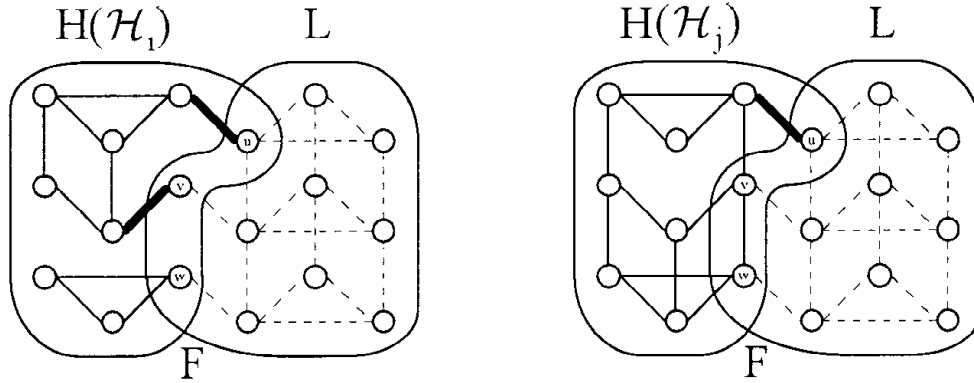


Fig. 5. State examples for 2ec reliability.

$L: \mathcal{H}_i \cup \mathcal{L}_j = \mathcal{G}_k$ . So the formula for the 2ec reliability of our graph can be written:

$$(8) \quad \mathcal{R}_{2ec}(G) = \sum_{\mathcal{H}_i, \mathcal{L}_j / H(\mathcal{H}_i) \cup L(\mathcal{L}_j) \text{ is 2-edge-connected}} \text{Prob}(\mathcal{H}_i) \cdot \text{Prob}(\mathcal{L}_j)$$

The subgraph  $H(\mathcal{H}_i)$  is composed of the set of the 2ec components  $\Phi(\mathcal{H}_i)$  and the set of the cut-edges  $\Delta(\mathcal{H}_i)$  that join these 2ec components,  $\Delta(\mathcal{H}_i) \subset \Phi(\mathcal{H}_i) \times \Phi(\mathcal{H}_i)$ , i.e., there exists an edge between two 2ec components  $\mathcal{T}_1$  and  $\mathcal{T}_2$  in  $\Delta(\mathcal{H}_i)$  if and only if there exists a cut-edge between  $u \in \mathcal{T}_1$  and  $v \in \mathcal{T}_2$  in  $H(\mathcal{H}_i)$ .  $(\Phi(\mathcal{H}_i), \Delta(\mathcal{H}_i))$  is a forest. We distinguish two kinds of 2ec components in  $\Phi(\mathcal{H}_i) = \Phi 1(\mathcal{H}_i) \cup \Phi 2(\mathcal{H}_i)$ :  $\Phi 1(\mathcal{H}_i)$  denotes the set of 2ec components of  $H(\mathcal{H}_i)$  that contain at least one vertex of  $F$  (i.e., the boundary 2ec components), and  $\Phi 2(\mathcal{H}_i)$  denotes the set of 2ec components of  $H(\mathcal{H}_i)$  that have no intersection with  $F$ .

To know if the result of  $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$  is 2-edge-connected, we have to consider  $\Phi(\mathcal{H}_i)$ ,  $\Phi(\mathcal{L}_j)$ ,  $\Delta(\mathcal{H}_i)$ , and  $\Delta(\mathcal{L}_j)$ .

We illustrate using the examples shown in Figure 5. The boundary set between  $H$  and  $L$  is  $F = \{u, v, w\}$ .

- $H(\mathcal{H}_j)$  contains two 2ec components:  $\Phi(\mathcal{H}_j) = \Phi 1(\mathcal{H}_j) = \{\mathcal{T}_{H_j,1}, \mathcal{T}_{H_j,2}\} (\mathcal{T}_{H_j,1} \cap F = \{u\}, \mathcal{T}_{H_j,2} \cap F = \{v, w\})$ , and the set  $\Delta(\mathcal{H}_j)$  contains a single cut-edge between these two components. All the vertices of  $H$  are connected to  $F$  by at least two paths, which is the condition for the state  $\mathcal{H}_j$  to be operating.
- $H(\mathcal{H}_i)$  contains four 2ec components:  $\Phi 1(\mathcal{H}_i) = \{\mathcal{T}_{H_i,1}, \mathcal{T}_{H_i,2}, \mathcal{T}_{H_i,3}\}$ ,  $\Phi 2(\mathcal{H}_i) = \{\mathcal{T}_{H_i,4}\} (\mathcal{T}_{H_i,1} \cap F = \{u\}, \mathcal{T}_{H_i,2} \cap F = \{v\}, \mathcal{T}_{H_i,3} \cap F = \{w\}, \mathcal{T}_{H_i,4} \cap F = \emptyset)$ , and two cut-edges in  $\Delta(\mathcal{H}_i)$  (between  $\mathcal{T}_{H_i,1}$  and  $\mathcal{T}_{H_i,4}$ , and between  $\mathcal{T}_{H_i,2}$  and  $\mathcal{T}_{H_i,4}$ ). All of the 2ec components are connected to  $F$  by at least two paths (so it is an operating state), nevertheless, the intersection between  $F$  and a component could be empty ( $\mathcal{T}_{H_i,4}$ ).

PROPOSITION 4.1. *The two following statements are equivalent:*

*Statement 1.  $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$  is 2-edge-connected.*



*Statement 2.*  $\mathcal{H}_i$  and  $\mathcal{L}_j$  are both operating states  
and  
the partial graph

$$\begin{aligned} & \bigcup_{\mathcal{T}_{H_i,x} \in \Phi 1(\mathcal{H}_i)} \{(F \cap \mathcal{T}_{H_i,x})\text{-}2\text{ec-clique}\} \\ & \cup \bigcup_{\mathcal{T}_{L_j,y} \in \Phi 1(\mathcal{L}_j)} \{(F \cap \mathcal{T}_{L_j,y})\text{-}2\text{ec-clique}\} \cup \Delta'(\mathcal{H}_i) \cup \Delta'(\mathcal{L}_j) \end{aligned}$$

is 2-edge-connected, where  $\Delta'(\mathcal{H}_i)$  is a transformation of  $\Delta(\mathcal{H}_i)$  to make possible this union, i.e., if an endpoint  $\mathcal{T}_{H_i,x}$  of an edge of  $\Delta(\mathcal{H}_i)$  belongs to  $\Phi 1(\mathcal{H}_i)$ , then this endpoint is replaced by a vertex of  $F \cap \mathcal{T}_{H_i,x}$ , hence  $\Delta'(\mathcal{H}_i) \subset (F \cup \Phi 2(\mathcal{H}_i)) \times (F \cup \Phi 2(\mathcal{H}_i))$ .

Proposition 4.1 leads to the following remark. As in the case of reliability, it is not necessary to know the identity of a vertex of  $H(\mathcal{H}_i)$  out of the boundary set in order to deduce that  $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$  is 2-edge-connected, because these vertices cannot be linked directly to a vertex of  $L$ . Hence, the information required for an operating state  $\mathcal{H}_i$  (to calculate (8)) is first the connection of the boundary vertices in 2ec components, and secondly the connections between these 2ec components. These paths, which can go through 2ec components of  $\Phi 2(\mathcal{H}_i)$ , are represented by the set of cut-edges  $\Delta(\mathcal{H}_i)$ . This information must be used to represent a class, but we reduce it below.

### 4.3. Class Definition and Equivalence Relation

4.3.1. *Introduction.* As in the case of reliability, the function of the 2ec reliability classes is to group all equivalent operating states, in order to factorize (8) with (5):

$$(9) \quad \mathcal{R}_{2\text{ec}}(G) = \sum_{C_{H,x}, C_{L,y} / C_{H,x} \text{ and } C_{L,y} \text{ are compatible}} \text{Prob}(C_{H,x}) \cdot \text{Prob}(C_{L,y}).$$

We also have here the failure classes  $\text{DEF}(H)$  and  $\text{DEF}(L)$  which group all failure states.

The information provided by a class must be just the information required to combine and find the compatible classes. Indeed, the more restricted the information standing for a class, the greater the number of equivalent states in this same class, and consequently the greater the efficiency of (9). We study in this section the information provided by a state to obtain the required information to define a class.

4.3.2. *State Representation.* The state  $\mathcal{H}_i$  can be represented as a forest of 2ec components:  $\mathcal{F}(\mathcal{H}_i) = (\Phi 1(\mathcal{H}_i) \cup \Phi 2(\mathcal{H}_i), \Delta(\mathcal{H}_i))$  where  $\Phi 1(\mathcal{H}_i)$  and  $\Phi 2(\mathcal{H}_i)$  are the sets of 2ec components defined in Section 4.2. We now reduce the knowledge provided by the forest  $\mathcal{F}(\mathcal{H}_i)$  to obtain a state representation denoted as  $\zeta(\mathcal{H}_i)$ . We saw with Proposition 4.1, that knowing the identities of the vertices outside of the boundary set is superfluous. So the state representation is a forest:  $\zeta(\mathcal{H}_i) = (\Theta 1(\mathcal{H}_i) \cup \Theta 2(\mathcal{H}_i), \Psi(\mathcal{H}_i))$  where  $\Theta 1(\mathcal{H}_i)$ ,  $\Theta 2(\mathcal{H}_i)$ , and  $\Psi(\mathcal{H}_i)$  correspond respectively to the reductions of  $\Phi 1(\mathcal{H}_i)$ ,  $\Phi 2(\mathcal{H}_i)$ , and  $\Delta(\mathcal{H}_i)$ .  $\Theta 1(\mathcal{H}_i)$  is a set of blocks of the boundary set  $F$ , that is to say a partition of  $F$ .  $\Theta 2(\mathcal{H}_i)$  is a set of empty blocks that we call unidentified blocks and  $\Psi(\mathcal{H}_i)$  is the set of edges joining these blocks (see Figure 6).

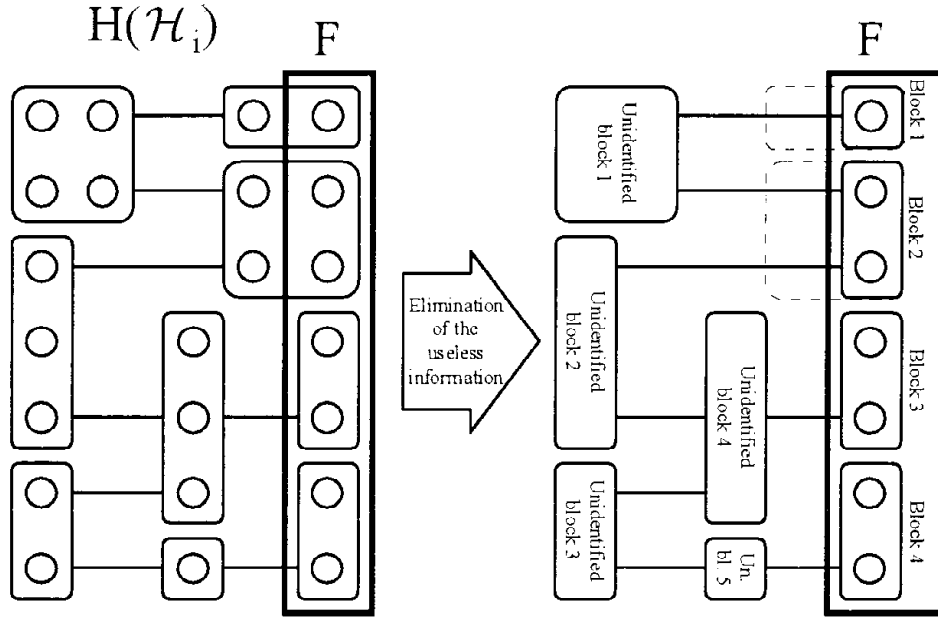


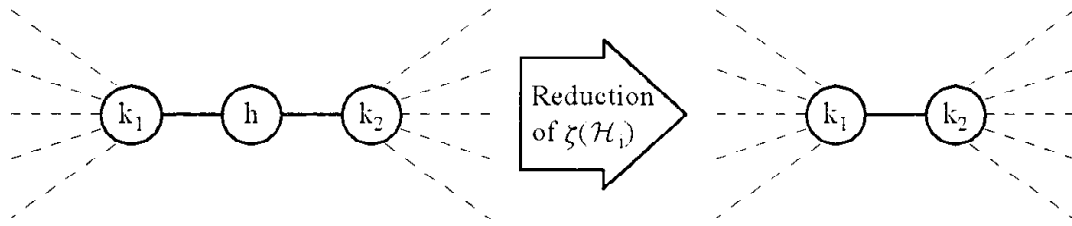
Fig. 6. State representation for 2ec reliability.  $H(\mathcal{H}_i)$  is a forest (here a tree) of nine 2ec components.

We use this representation for the examples of Figure 5:

$$\begin{aligned}
 \zeta(\mathcal{H}_i) &= (\Theta 1(\mathcal{H}_i) \cup \Theta 2(\mathcal{H}_i), \Psi(\mathcal{H}_i)) \\
 &\text{with } \Theta 1(\mathcal{H}_i) = \{B_1, B_2, B_3\}, \quad \Theta 2(\mathcal{H}_i) = \{B_{-1}\}, \\
 &\quad \Psi(\mathcal{H}_i) = \{(B_1, B_{-1}), (B_2, B_{-1})\}, \\
 &\text{and } B_1 = [u], \quad B_2 = [v], \quad B_3 = [w], \quad B_{-1} = []; \\
 \zeta(\mathcal{H}_j) &= (\Theta 1(\mathcal{H}_j) \cup \Theta 2(\mathcal{H}_j), \Psi(\mathcal{H}_j)) \\
 &\text{with } \Theta 1(\mathcal{H}_j) = \{B_1, B_2\}, \quad \Theta 2(\mathcal{H}_j) = \{\}, \quad \Psi(\mathcal{H}_j) = \{(B_1, B_2)\}, \\
 &\text{and } B_1 = [u], \quad B_2 = [vw].
 \end{aligned}$$

With such a state representation ( $\zeta(\mathcal{H}_i)$ ), the composition of the 2ec components of  $\Phi 2(\mathcal{H}_i)$ , represented by the unidentified blocks of  $\Theta 2(\mathcal{H}_i)$ , is totally unknown (whereas the composition of a boundary 2ec component is partially known with  $\Theta 1(\mathcal{H}_i)$ ). This notion of unidentified blocks allows us to represent all the required connection manners of the boundary blocks of the partition  $\Theta 1(\mathcal{H}_i)$ . The single condition for an unidentified block to exist is the presence of edges in  $\Psi(\mathcal{H}_j)$  incident to this unidentified block, which represents useful block path information. In the following, we eliminate some unidentified blocks without deleting this essential information.

4.3.3. *From State Representation to Minimal State Representation.* We now remove the useless information of  $\zeta(\mathcal{H}_i)$  and obtain the minimal state representation denoted as  $\zeta'(\mathcal{H}_i)$ , i.e., the bare necessities. A simplification can be made for  $\Theta 2(\mathcal{H}_i)$ . Indeed, if there is an unidentified block  $h \in \Theta 2(\mathcal{H}_i)$  which is adjacent to only two other blocks,  $(h, k_1) \in \Psi(\mathcal{H}_i)$ ,  $(h, k_2) \in \Psi(\mathcal{H}_i)$ , then we need retain no information other than the path between  $k_1$  and  $k_2$ , i.e.,  $(k_1, k_2)$  can replace  $(h, k_1)$  and  $(h, k_2)$  in  $\Psi(\mathcal{H}_i)$  enabling



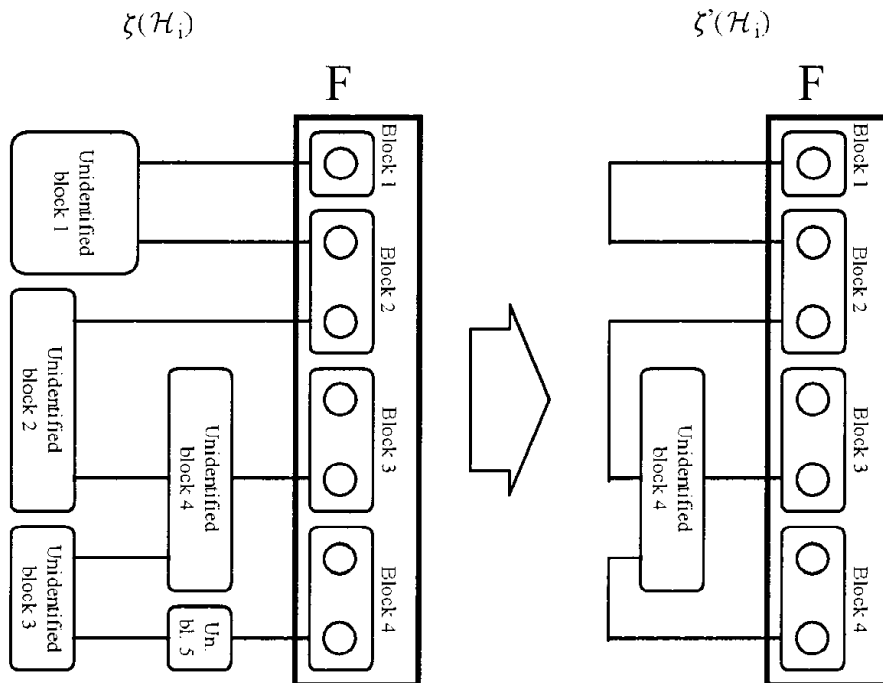
**Fig. 7.** From state representation to minimal state representation.  $h$  is an unidentified block of degree two.

the unidentified block  $h$  to be removed from  $\Theta 2(\mathcal{H}_i)$  (Figure 7). This is a reduction of  $\Theta 2(\mathcal{H}_i)$  to  $\Theta 2'(\mathcal{H}_i)$ , and a transformation from  $\Psi(\mathcal{H}_i)$  to  $\Psi'(\mathcal{H}_i)$ .

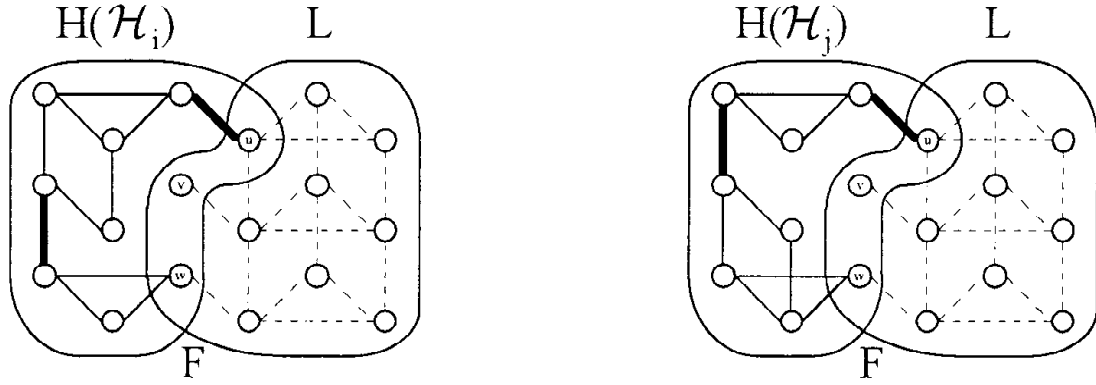
After the deletion of all the unidentified blocks of degree two in  $\Psi(\mathcal{H}_i)$  and  $\Theta 2(\mathcal{H}_i)$ , we obtain  $\Psi'(\mathcal{H}_i)$  and  $\Theta 2'(\mathcal{H}_i)$ , and no further simplification of the information for a state is possible (see Figure 8). The minimal representation of a state is composed of a forest whose nodes are blocks of a partition and unidentified blocks:  $\zeta'(\mathcal{H}_i) = (\Theta 1(\mathcal{H}_i) \cup \Theta 2'(\mathcal{H}_i), \Psi'(\mathcal{H}_i))$ , such that the unidentified blocks have a degree strictly larger than two. Indeed it must be at least two in order to have an operating state, and it cannot be two because of the reduction from  $\Psi(\mathcal{H}_i)$  to  $\Psi'(\mathcal{H}_i)$ .

**PROPERTY 4.2.** For any unidentified block  $b$  of  $\Theta 2'(\mathcal{H}_i)$ , the degree of  $b$  in the forest  $\zeta'(\mathcal{H}_i) = (\Theta 1(\mathcal{H}_i) \cup \Theta 2'(\mathcal{H}_i), \Psi'(\mathcal{H}_i))$  is strictly larger than two.

Consequently, we have  $0 \leq |\Theta 2'(\mathcal{H}_i)| \leq \max(0, |\Theta 1(\mathcal{H}_i)| - 2)$ .



**Fig. 8.** From state representation to minimal state representation (example). Removal of the unidentified blocks of degree two.



**Fig. 9.** Equivalent states for 2ec reliability.  $\zeta(\mathcal{H}_i) = (\Theta 1(\mathcal{H}_i) \cup \Theta 2(\mathcal{H}_i), \Psi(\mathcal{H}_i)) = (\{B_1, B_2, B_3, B_{-1}\}, \{(B_1, B_{-1}), (B_3, B_{-1})\})$  with  $B_1 = [u], B_2 = [v], B_3 = [w], B_{-1} = []$ .  $\zeta(\mathcal{H}_j) = (\Theta 1(\mathcal{H}_j) \cup \Theta 2(\mathcal{H}_j), \Psi(\mathcal{H}_j)) = \zeta(\mathcal{H}_i) \Rightarrow \mathcal{H}_i$  and  $\mathcal{H}_j$  are equivalent states.

**4.3.4. Class Representation.** Several states have the same minimal representation. These states are equivalent and belong to the same class (see Figure 9). The modeling of a class  $C_{H,k}$  is therefore a forest  $C_{H,k} = (\Theta 1(C_{H,k}) \cup \Theta 2'(C_{H,k}), \Psi'(C_{H,k}))$ , where:

- $\Theta 1(C_{H,k})$  is a partition of the boundary set  $F$  of  $|\Theta 1(C_{H,k})|$  blocks.
- $\Theta 2'(C_{H,k})$  is a set of unidentified blocks which satisfy Property 4.2.
- $\Psi'(C_{H,k})$  is a set of edges whose endpoints belong to  $\Theta 1(C_{H,k}) \cup \Theta 2'(C_{H,k})$ .

All the states whose minimal representation is  $C_{H,k}$  belong to this class.

**4.4. The Decomposition Principle.** The principle of decomposition for 2ec reliability is similar to that for all terminal reliability:

- consider two subgraphs  $H$  and  $L$  and their boundary set  $F$ ,
- enumerate the operating classes of  $H$  and  $L$  (omitting the failure classes  $\text{DEF}(H)$  and  $\text{DEF}(L)$ ),
- compute the associated probabilities of these classes (with (5)),
- search for all compatible classes and compute the 2ec reliability with (9).

2ec reliability of  $G$  is computed by combining the compatible classes of  $H$  and  $L$  (9). Two classes  $C_{H,x}$  and  $C_{L,y}$  are compatible if their union provides the 2-edge-connectivity of the boundary set  $F$ , and can ensure the 2-edge-connectivity of the whole graph  $G$ .

Figure 10 presents the set of possible classes for a boundary set of three vertices. Note: the blocks of the partitions are denoted  $B_i$  with  $i > 0$  and the unidentified blocks are denoted  $B_j$  with  $j < 0$ . We list here some combinations between the classes of the two subgraphs  $H$  and  $L$ , separated by a boundary set of three vertices, that are compatible

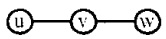
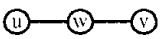
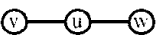


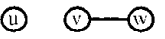

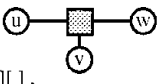
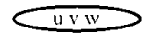


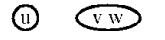
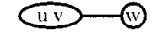
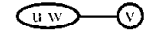
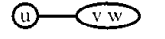
 ( [u][v][w] , { (B <sub>1</sub> ,B <sub>2</sub> ) , (B <sub>2</sub> ,B <sub>3</sub> ) } )	 ( [u][v][w] , { (B <sub>1</sub> ,B <sub>3</sub> ) , (B <sub>2</sub> ,B <sub>3</sub> ) } )	 ( [u][v][w] , { (B <sub>1</sub> ,B <sub>2</sub> ) , (B <sub>1</sub> ,B <sub>3</sub> ) } )
 ( [u][v][w] , { (B <sub>1</sub> ,B <sub>2</sub> ) } )	 ( [u][v][w] , { (B <sub>1</sub> ,B <sub>3</sub> ) } )	 ( [u][v][w] , { (B <sub>2</sub> ,B <sub>3</sub> ) } )
 ( [u][v][w] , { } )	 ( [u][v][w][ ] , { (B <sub>1</sub> ,B <sub>1</sub> ), (B <sub>1</sub> ,B <sub>2</sub> ), (B <sub>1</sub> ,B <sub>3</sub> ) } )	 ( [uvw] , { } )
 ( [uv][w] , { } )	 ( [uw][v] , { } )	 ( [u][vw] , { } )
 ( [uv][w] , { (B <sub>1</sub> ,B <sub>2</sub> ) } )	 ( [uw][v] , { (B <sub>1</sub> ,B <sub>2</sub> ) } )	 ( [u][vw] , { (B <sub>1</sub> ,B <sub>2</sub> ) } )

Fig. 10. The 15 possible classes for a boundary set of three vertices.

for 2ec reliability:

$$\begin{aligned}
 C_{H,1} &= ([uvw], \{ \}) & \text{and } C_{L,i} &, \forall i, 1 \leq i \leq 15. \\
 C_{H,3} &= ([uv][w], \{(1, 2)\}) & \text{and } C_{L,3} &= ([uv][w], \{(1, 2)\}). \\
 C_{H,5} &= ([uw][v], \{(1, 2)\}) & \text{and } C_{L,3} &= ([uv][w], \{(1, 2)\}). \\
 C_{H,5} &= ([uw][v], \{(1, 2)\}) & \text{and } C_{L,6} &= ([u][vw], \{ \}). \\
 C_{H,6} &= ([u][vw], \{ \}) & \text{and } C_{L,1} &= ([uvw], \{ \}). \\
 C_{H,8} &= ([u][v][w], \{ \}) & \text{and } C_{L,1} &= ([uvw], \{ \}). \\
 C_{H,12} &= ([u][v][w], \{(1, 2), (1, 3)\}) & \text{and } C_{L,14} &= ([u][v][w], \{(1, 3), (2, 3)\}). \\
 C_{H,12} &= ([u][v][w], \{(1, 2), (1, 3)\}) & \text{and } C_{L,15} &= ([u][v][w][ ], \{(-1, 1), (-1, 2), (-1, 3)\}). \\
 C_{H,15} &= ([u][v][w][ ] \{(-1, 1), (-1, 2), (-1, 3)\}) & \text{and } C_{L,15} &= ([u][v][w][ ], \{(-1, 1), (-1, 2), (-1, 3)\}).
 \end{aligned}$$

Rosenthal’s algorithm described in Section 3.4 for all terminal reliability can be applied for 2ec reliability without any change.

4.5. *Number of Classes.* The number of classes depends on the number of vertices in the boundary set:  $|F|$  (e.g., the class enumeration for  $|F| = 3$  is shown in Figure 10). A class is composed of a partition of  $F$  with  $k$  blocks and of a forest on these blocks, and possibly of unidentified blocks of degree strictly greater than two (see Property 4.2). So the number of classes for a given boundary set  $F$  is

$$\text{Nb\_classes}_{2ec} = \sum_{j=1}^{|F|} (A_{|F|,j} \cdot \text{NFU}(j)),$$

where  $A_{|F|,j}$  is the Stirling number of the second kind (see Section 3.5) and  $\text{NFU}(j)$  stands for the number of these particular labeled forests of  $j$  nodes with unidentified nodes (these unidentified nodes are not labeled and have a degree strictly greater than two). These numbers are given in Tables 2 and 3.

**Table 2.** Number of labeled forests with unidentified nodes against the number of nodes  $N$ .

$N$	1	2	3	4	5	6	7
NFU( $N$ )	1	2	8	58	662	10,584	219,004

**5. Linear Time Algorithms.** We have seen above the decomposition principle which consists in considering two subgraphs  $H$  and  $L$  of  $G$  and combining their classes to compute reliability (see Sections 3 and 4). Now we look at the implementation of this principle with an algorithm belonging to the table-based reduction algorithm family [12], more effective than Rosenthal's algorithm which we presented in Section 3.4.

**5.1. Preliminary Definitions.** We first define the notions of linear ordering and vertex separation number used in our algorithm, and the notions of pathwidth and treewidth introduced by Robertson and Seymour [13], [14] and Bodlaender [15].

By definition, a **linear ordering** (denoted  $\mathcal{N}$ ) of  $G = (V, E)$  is a bijection:  $\mathcal{N}: V \rightarrow \{1, \dots, |V|\}$ .

In the following, the vertex  $\mathcal{N}^{-1}(i)$ ,  $i \in \{1, \dots, |V|\}$ , will be denoted as  $v_i$ .

We denote  $F_i = \{v_j \in V / \exists (v_j, v_k) \in E \text{ such that } j \leq i \leq k\}$ ,  $\forall i \in \{1, \dots, |V|\}$ .

The **vertex separation number** (denoted  $F_{\max}$ ) of a linear ordering is  $F_{\max}(\mathcal{N}) = \max_{1 \leq i \leq |V|} (|F_i|)$ .

The vertex separation number of a graph is  $F_{\max}(G) = \min_{\mathcal{N}} (F_{\max}(\mathcal{N}))$ .

A **path-decomposition** (denoted as  $\mathcal{D}_p$ ) of  $G = (V, E)$  is a sequence of subsets of  $V$ :  $\mathcal{D}_p = (X_1, \dots, X_r)$ , such that:

- $\cup_{1 \leq i \leq r} X_i = V$ ,
- for every edge  $(v, w) \in E$ , there is a subset  $X_i$ ,  $1 \leq i \leq r$ , with  $v \in X_i$  and  $w \in X_i$ ,
- for all  $i, j, k \in \{1, \dots, r\}$ , if  $i \leq j \leq k$ , then  $X_i \cap X_k \subseteq X_j$ .

By definition, the **pathwidth** of a path-decomposition is  $\text{pathwidth}(\mathcal{D}_p) = \max_{1 \leq i \leq r} (|X_i| - 1)$ , and the pathwidth of a graph is  $\text{pathwidth}(G) = \min_{\mathcal{D}_p} (\text{pathwidth}(\mathcal{D}_p))$ .

A **tree-decomposition** (denoted as  $\mathcal{D}_t$ ) of  $G = (V, E)$  is a couple composed of a family of subsets of  $V$  and a tree:  $\mathcal{D}_t = (\{X_i / i \in I\}, T = (I, F))$ , such that:

- $\cup_{i \in I} X_i = V$ ,
- for every edge  $(v, w) \in E$ , there is a subset  $X_i$ ,  $i \in I$ , with  $v \in X_i$  and  $w \in X_i$ ,
- for all  $i, j, k \in I$ , if  $j$  is on the path from  $i$  to  $k$  in  $T$ , then  $X_i \cap X_k \subseteq X_j$ .

**Table 3.** Total number of classes (2ec reliability) against the size of  $F$ .

$ F $	1	2	3	4	5	6	7
Nb_classes <sub>2ec</sub>	1	3	15	121	1473	25,067	556,783

By definition, the **treewidth** of a tree-decomposition is  $\text{treewidth}(\mathcal{D}_t) = \max_{i \in I} (|X_i| - 1)$ , and the treewidth of a graph is  $\text{treewidth}(G) = \min_{\mathcal{D}_t} (\text{treewidth}(\mathcal{D}_t))$ .

The following propositions have been demonstrated: Let  $G = (V, E)$  be a given graph. The pathwidth of a graph  $G$  is at least its treewidth. In fact, a path-decomposition of  $G$  can be written as a tree-decomposition of  $G$ , with the same width. The pathwidth of a graph  $G$  is equal to its vertex separation number [16]. Moreover, finding a linear ordering of  $G$  with a minimum vertex separation number is equivalent to finding a path-decomposition with the smallest pathwidth.

The pathwidth and treewidth problems (given a graph, find a tree-decomposition or a path-decomposition with the smallest width) are NP-hard [17]. Nevertheless, for fixed parameters  $k$ , the problems of finding a path-decomposition or tree-decomposition of width at most  $k$  can be solved in linear time, but using algorithms with a rather high constant factor [18], [19].

*5.2. The Dynamic Programming Algorithm for Reliability and 2ec Reliability Computations.* We now present our algorithm based on the decomposition principle (see Sections 3 and 4). We consider a resolved subgraph  $H$  whose classes are known, and we enlarge this resolved subgraph by vertex insertion until we have resolved the whole graph.

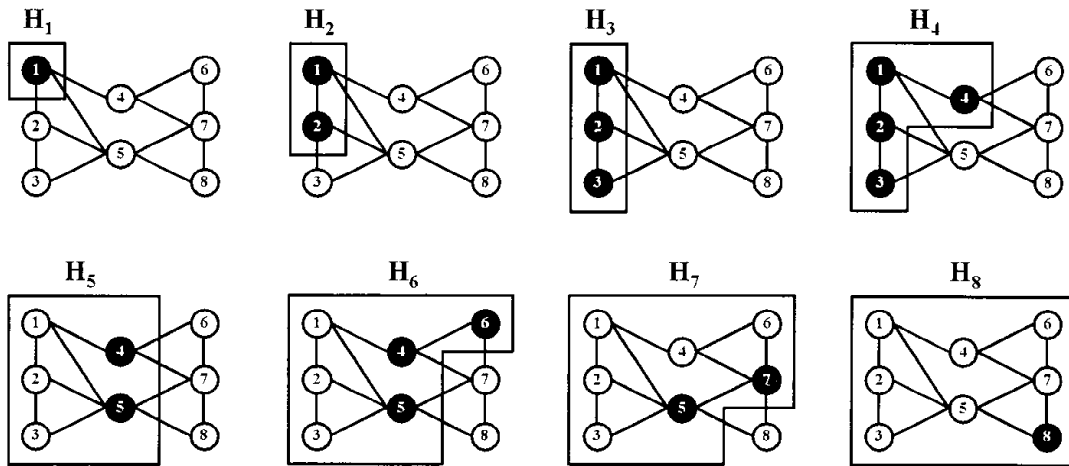
0. A linear ordering of the vertices is required.
1.  $H_0 = \{\}, L_0 = G$ .
2. For each vertex  $v_i$  (in the order given by the linear ordering, from  $v_1$  to  $v_{|V|}$ ):
  - 2.1. Remove  $v_i$  from the subgraph  $L_{i-1}$  and add  $v_i$  to the subgraph  $H_{i-1}$  to obtain  $L_i$  and  $H_i$ .
  - 2.2. Find the boundary set  $F_i$  between  $H_i$  and  $L_i$ .
  - 2.3. Compute the classes of the boundary set  $F_i$  in the subgraph  $H_i$  and their probabilities.
3. The reliability (or 2ec reliability) of  $G$  is the probability of the single class of the last boundary set.

The vertex separation number,  $F_{\max}$ , is the size of the boundary set  $F_i$  that contains the maximal number of vertices during the algorithm.

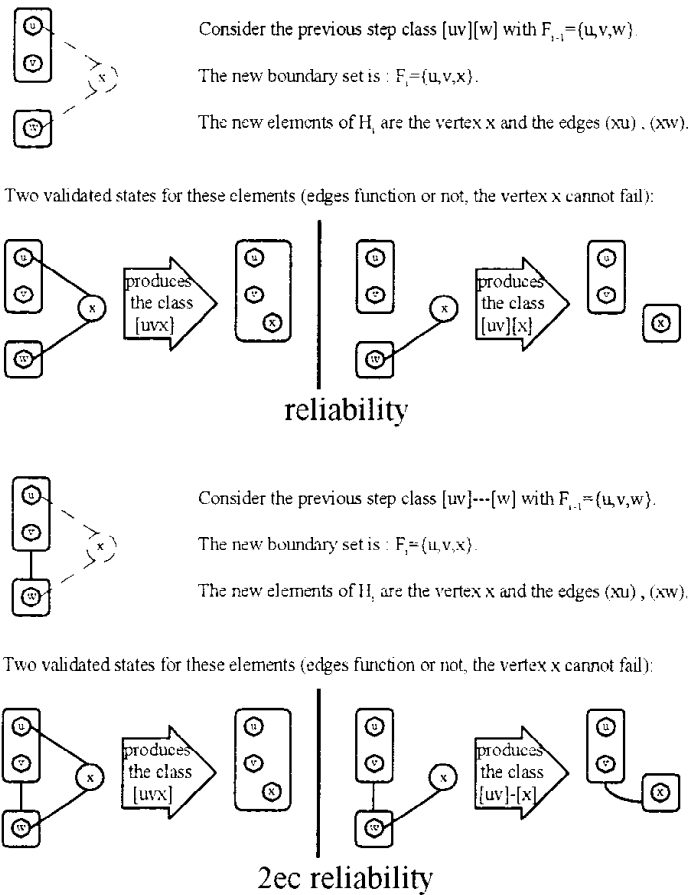
Figure 11 shows the boundary set evolution and the growth of the resolved subgraph ( $H_i$ ) during the algorithm. At each step of the algorithm, the classes and their probabilities are stored in a table. The table of each step is computed using the table of the previous one. For each class we need to enumerate the possible states of the new elements introduced in the resolved graph  $H_i$ , i.e., the new vertex and the new edges added to the new resolved subgraph  $H_i$  (see Figure 12).

### 5.3. Complexity and Vertex Separation Number

*5.3.1. Complexity of All Terminal Reliability Computation.* The results of this algorithm [11] prove that the decomposition method is extremely powerful. In practice, for the all terminal reliability problem, it can handle any network with  $F_{\max} \leq 12$ . Indeed, the resolution time and required memory grow exponentially in function of  $F_{\max}$ . The



**Fig. 11.** Evolution of the resolved subgraph during the algorithm. The vertices belonging to  $F_i$  are shaded.  $F_{\max} = 4$ .



**Fig. 12.** Construction of the new step classes.



**Table 4.** Reliability computation times (in seconds) with the decomposition algorithm.

	$ V $									
	10	20	30	40	50	60	70	80	90	100
$F_{\max} = 6$	0.18	0.66	1.16	1.66	2.16	2.64	3.12	3.66	4.12	4.62
$F_{\max} = 7$	0.87	4.23	7.61	10.98	14.39	17.90	21.20	24.79	27.94	31.40
$F_{\max} = 8$	3.36	28.57	53.45	79.08	103.94	128.77	154.09	179.12	204.30	230.21
$F_{\max} = 9$	5.75	202.53	399.74	597.73	793.49	991.21	1186.84	1384.47	1585.97	1775.92

complexity is

$$O\left(|V| \cdot \sum_{j=1}^{F_{\max}} (A_{F_{\max},j} \cdot 2^j) \cdot (F_{\max})^2\right),$$

where  $A_{i,j}$  is the Stirling number of the second kind, and  $F_{\max}$  is the vertex separation number. We now give a brief explanation of this complexity:  $|V|$  stands for the number of main iterations, i.e., the insertion of a new vertex  $v_i$  in  $H_i$ .  $\sum A_{F_{\max},j}$  is the number of possible classes for a boundary set of size  $F_{\max}$ .  $2^j$  represents the number of possible states to consider for the edges added to  $H_i$ .  $(F_{\max})^2$  stands for the class construction and identification in the new resolved subgraph  $H_{i+1}$ .

The exponential factor of this method is  $F_{\max}$ , whereas this factor is the size of the graph for the factoring-reduction method [9], i.e., this algorithm can compute the reliability of large networks whereas the factoring-reduction algorithm is limited to small networks.

Table 4 presents the results of the decomposition algorithm in CPU time, running on a 167 megahertz machine (UltraSparc), for large density graphs with 10–100 vertices, for  $F_{\max}$  from 6 to 9. These graphs are  $F_{\max}$ -paths, which means that the addition of an edge in such a graph increases its  $F_{\max}$ .

**5.3.2. Complexity of 2ec Reliability Computation.** As in the case of reliability, for all terminal 2ec reliability we use a table-based reduction algorithm. The 2ec reliability algorithm requires more classes than the reliability algorithm, which explains why  $F_{\max} \leq 7$  in practice. Moreover, these classes are more costly to compute and handle in memory.

The complexity of this algorithm is

$$O\left(|V| \cdot \sum_{j=1}^{F_{\max}} (A_{F_{\max},j} \cdot \text{NFU}(j)) \cdot 3^j \cdot (F_{\max}) \cdot \text{NFU}(F_{\max})\right),$$

where  $A_{i,j}$  is the Stirling number of the second kind,  $F_{\max}$  the vertex separation number, and  $\text{NFU}(j)$  is the number of labeled forests with  $j$  nodes and  $k$  unidentified nodes with  $0 \leq k \leq \max(0, j - 2)$  (see Property 4.2). We now give a brief explanation for this complexity:  $|V|$  stands for the number of main iterations, i.e., the insertion of a new vertex  $v_i$  in  $H_i$ .  $\sum (A_{F_{\max},j} \cdot \text{NFU}(j))$  is the number of possible classes for a boundary set of size  $F_{\max}$ .  $3^j$  represents the number of possible states to consider for the edges added to  $H_i$ .  $F_{\max} \cdot \text{NFU}(F_{\max})$  stands for the class construction and identification in the new resolved subgraph  $H_{i+1}$ .

**Table 5.** 2ec reliability computation times (in seconds) with the decomposition algorithm.

	$ V $									
	10	20	30	40	50	60	70	80	90	100
$F_{\max} = 5$	7.60	30.31	53.18	76.09	97.54	158.29	141.41	164.54	197.85	209.47
$F_{\max} = 6$	182.16	1306.62	2336.50	3491.10	4560.01	5711.74	6634.53	7703.86	8769.99	9857.80

Table 5 presents the results of the decomposition algorithm in CPU time, running on a 167 megahertz machine (UltraSparc), for large density graphs with 10–100 vertices, for  $F_{\max} = 5$  and 6. These graphs are  $F_{\max}$ -paths (see Section 5.3.1).

Table 6 presents our results in CPU time, running on a 100 megahertz machine, for small density graphs (grid networks) with  $3 \times 2$  to  $3 \times 9$  vertices, for  $F_{\max} = 3$ , and compares them with an algorithm using a classical state enumeration.

**5.3.3. Linear Time and Vertex Separation Number.** We have seen in Sections 5.3.1 and 5.3.2 that the main factor of the complexity for the all terminal reliability and 2ec reliability algorithms is  $F_{\max}$ , the size of the maximum boundary set. Moreover, for a given bounded  $F_{\max}$ , the complexity is linear in  $O(|V|)$  (all other factors of the complexity are constant). The linearity of our results can be verified in Tables 4 and 5.

The largest size of the boundary set during our algorithms,  $F_{\max}$ , depends on the initial vertex linear ordering and so does the complexity. Figure 13 shows another linear ordering for the graph of Figure 11, with  $F_{\max} = 3$  instead of  $F_{\max} = 4$ .

A basic problem is therefore to find a linear ordering for a given graph such that  $F_{\max}$  is minimal. This problem is equivalent to the problem of the pathwidth (see Section 5.1). To solve it, we use a heuristic method described in [11].

**5.4. Dynamic Programming Algorithms to Solve NP-Hard Problems.** Our algorithm can solve the network reliability problem, which is NP-hard, in linear time for a bounded  $F_{\max}$ . It has been proved that some classes of graph problems can be solved in polynomial (or linear time) with such dynamic programming algorithms using a tree-decomposition with a bounded width [20]–[23]. The principle of these algorithms is to use the graph tree topology in order to expand a resolved subgraph, until the whole graph is resolved, and to store in memory all partial solutions standing for the resolved elements, i.e., the information required to compute the final solution.

The algorithm presented in this article (Section 5.2) uses a path-decomposition (which is a particular tree-decomposition) produced by the linear ordering of the vertices. The

**Table 6.** 2ec reliability computation times (in seconds) with the decomposition method (DEC) and with an enumeration method (ENU).

	$ V $							
	6	9	12	15	18	21	24	27
DEC	0.84	0.80	0.76	0.95	0.65	0.67	0.67	0.66
ENU	0.01	0.03	0.14	1.12	9.24	77.59	651.87	5452.46

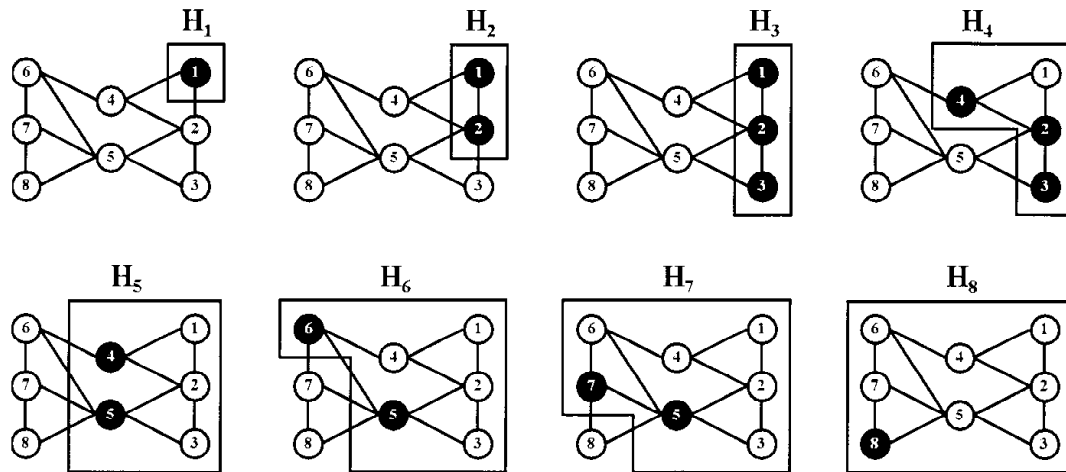


Fig. 13. Another vertex linear ordering. The vertices belonging to  $F_i$  are shaded.  $F_{\max} = 3$ .

vertex separation number ( $F_{\max}$ ) of the linear ordering corresponds to the pathwidth of the path-decomposition [16].

**6. Conclusion.** We have presented algorithms to compute all terminal reliability and 2ec reliability in linear time for graphs with bounded pathwidths. These algorithms belong to the family of dynamic programming algorithms which solve some NP-hard problems in polynomial (or linear time) with a given small width tree-decomposition. The implementation of our algorithms uses a vertex linear ordering of the graph with a vertex separation number equivalent to the correspondent pathwidth. For greater efficiency, a similar algorithm could be implemented using a tree-decomposition instead of a path-decomposition.

The main difficulty of such dynamic programming algorithms for a given problem is to find and handle the classes. Here we have described the definition of these classes for all terminal reliability and all terminal 2ec reliability with perfect vertices. The classes for  $K$ -terminal reliability with imperfect vertices are defined in [11]. With regard to 2-connected reliability, the huge number of classes given in [24] does not allow the decomposition method to be as effective as with reliability and 2ec reliability.

## References

- [1] A. Rosenthal, Computing the reliability of complex networks, *SIAM J. Appl. Math.*, **32** (1977), 384–393.
- [2] C. J. Colbourn and D. D. Harms, Bounding all terminal reliability in computer networks, *Networks*, **18** (1988), 1–12.
- [3] M. O. Ball, Computing network reliability, *Oper. Res.*, **27** (1979), 823–838.
- [4] O. Frank and W. Gaul, On reliability in stochastic graphs, *Networks*, **12** (1982), 119–126.
- [5] O. Theologou, Contribution à l'évaluation de la fiabilité des réseaux, Ph.D. Thesis, Université de technologie de Compiègne, 1990.
- [6] M. O. Ball, Complexity of network reliability computation, *Networks*, **10** (1980), 153–165.
- [7] A. Satyanarayana and M. Chang, Network reliability and the factoring theorem, *Networks*, **13** (1983), 107–120.

- [8] R. K. Wood, A factoring algorithm using polygon-to-chain reductions for computing  $K$ -terminal network reliability, *Networks*, **15** (1985), 173–190.
- [9] O. Theologou and J. Carlier, Factoring and reductions for networks with imperfect vertices, *IEEE Trans. Reliability*, **40** (1991), 210–217.
- [10] J. Carlier and C. Lucet, A decomposition algorithm for network reliability evaluation, *Discrete Appl. Math.*, **65** (1996), 141–156.
- [11] C. Lucet, Méthode de décomposition pour l'évaluation de la fiabilité des réseaux, Ph.D. Thesis, Université de technologie de Compiègne, 1993.
- [12] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey, *BIT*, **25** (1985), 2–23.
- [13] N. Robertson and P. D. Seymour, Graph minors. I. Excluding a forest, *J. Combin. Theory Ser. B*, **35** (1983), 39–61.
- [14] N. Robertson and P. D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *J. Algorithms*, **7** (1986), 309–322.
- [15] H. L. Bodlaender, A tourist guide through treewidth, *Acta Cybernet.*, **11** (1993), 1–23.
- [16] N. G. Kinnersley, The vertex separation number of a graph equals its path width, *Inform. Process Lett.*, **42** (1992), 345–350.
- [17] S. Arnborg, D. G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a  $k$ -tree, *SIAM J. Algebraic Discrete Methods*, **8** (1987), 277–284.
- [18] H. L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.*, **25** (1996), 1305–1317.
- [19] H. L. Bodlaender and T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, *J. Algorithms*, **21** (1996), 358–402.
- [20] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms*, **12** (1991), 308–340.
- [21] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems on graphs embedded in  $k$ -trees, *Discrete Appl. Math.*, **23** (1989), 11–24.
- [22] H. L. Bodlaender, Dynamic programming on graphs with bounded treewidth, *Proc. 15th ICALP '88*, LNCS 317, Springer-Verlag, Berlin, 1988, pp. 105–118.
- [23] B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, *Theoret. Comput. Sci.*, **109** (1993), 49–82.
- [24] J. F. Manouvrier, Méthode de décomposition pour résoudre des problèmes combinatoires sur les graphes, Ph.D. Thesis, Université de technologie de Compiègne, 1998.

# K-Terminal Network Reliability Measures With Binary Decision Diagrams

Gary Hardy, Corinne Lucet, and Nikolaos Limnios

**Abstract**—We present a network decomposition method using Binary Decision Diagrams (BDD), a state-of-the-art data structure to encode, and manipulate Boolean functions, for computing the reliability of networks such as computer, communication, or power networks. We consider the  $K$ -terminal reliability measure  $R_K$ , which is defined as the probability that a subset  $K$  of nodes can communicate with each other, taking into account the possible failures of the network links. We present an exact algorithm for computing the  $K$ -terminal reliability of a network with perfect vertices in  $O(m \cdot F_{max} \cdot 2^{F_{max}} \cdot B_{F_{max}})$ , where  $B_{F_{max}}$  is the Bell number of the maximum boundary set of vertices  $F_{max}$ , and  $m$  is the number of network links. Several examples, and experiments show the effectiveness of this approach.

**Index Terms**—Binary decision diagram, network reliability computation.

## ACRONYM<sup>1</sup>

BDD Binary Decision Diagram  
DAG Directed Acyclic Graph

## NOTATION

$G$   $(V, E)$ : network model  
 $V$  vertex set of  $G$   
 $E$   $\{e_1, \dots, e_m\}$ : edge set of  $G$   
 $m$  number of links ( $m = |E|$ )  
 $K$   $K \subseteq E$ ; terminal nodes of  $G$   
 $x_i$  state of link  $e_i$   
 $\bar{x}_i$   $\bar{x}_i = 1 - x_i$   
 $x$   $(x_1, \dots, x_m)$ : state vector of  $G$   
 $p_i$   $Pr(x_i = 1)$ : functioning probability of  $e_i$   
 $q_i$   $q_i = 1 - p_i = Pr(x_i = 0)$   
 $R_K$   $K$ -terminal network reliability  
 $B_n$  the  $n$ th Bell number

## I. INTRODUCTION

NETWORK reliability analysis receives considerable attention for the design, validation, and maintenance of many real world systems, such as computer, communication, or power

networks. The reliability of these complex systems is an increasing concern as the failure of some of their components could lead to disastrous results. The IEEE 90 standard [1] defines the reliability as “the ability of a system or component to perform its required functions under stated conditions for a specified period of time.” In this paper, we take an interest in the connection functions of networks, and we propose an algorithm based on *Binary Decision Diagrams* [2], [3] for computing the network reliability.

Two kinds of computations exist for the network reliability: approximate, and exact. Our algorithm provides an exact network reliability value. The network model is an undirected stochastic graph  $G = (V, E)$ , where  $V$  is the vertex set, and  $E$  is the edge set. Sites (workstations, routers, etc.) correspond to vertices, and links between sites (fiber-optic, electric cable, etc.) correspond to edges. Each edge, and each vertex can fail randomly, and independently with known probability. In real problems, these probabilities are usually computed from statistical data. The connection function concerns all, or a subset of vertices. The more general one is the  $K$ -terminal connection function that consists of ensuring the connection between all pairs of terminal vertices of subset  $K$ . This problem is well-known to be NP-hard [4], [5]. Provan [11] showed that even for planar graphs this problem is still NP-hard.

In literature, two classes of exact methods are often used to compute the network reliability. The first class deals with the enumeration of all the minimum paths or cuts. A *path* is defined as a set of network components (edges and/or vertices) so that if these components are all failure-free, the system is up. A path is *minimal* if it has no proper subpaths. Conversely, a *cut* is a set of network components such that if these components fail, the system is down. The probabilistic evaluation uses the *inclusion-exclusion*, or *sum of disjoint products* methods because this enumeration provides non-disjoint events. Numerous works about this kind of methods have been presented in literature, [14]–[16]. Such methods cannot treat large networks (up to 20 vertices with low density).

In the second class, the algorithms are based on graph topology. The first process consists of reducing the graph by replacing some special substructures by smaller ones. The replaced substructures can be elementary, such as two adjacent edges (*series-parallel reductions*), or more complex as a subgraph (*polygon-to-chain* [23], and *delta-to-star reductions* [24]). These reductions allow us to compute the reliability of series-parallel networks in linear time; the reductions are recursively applied until resulting in a single edge. Nevertheless, for general networks, such substructures cannot be found. In this case, the factoring process is applied. The idea is to choose

Manuscript received March 20, 2006; revised October 30, 2006; accepted December 2, 2006. This work was supported by the Conseil Regional de Picardie. Associate Editor: R. H. Yeh

G. Hardy and C. Lucet are with the LaRIA, Université de Picardie Jules Verne, France (e-mail: gary.hardy@u-picardie.fr; corinne.lucet@u-picardie.fr).

N. Limnios is with the LMAC, Université de Technologie de Compiègne, France (e-mail: nikolaos.limnios@utc.fr).

Digital Object Identifier 10.1109/TR.2007.898572

<sup>1</sup>The singular and plural of an acronym are always spelled the same.

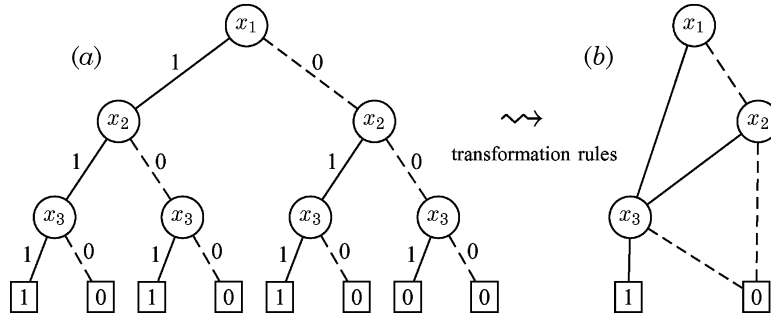


Fig. 1. (a) Decision tree of  $(x_1 \wedge x_3) \vee (x_2 \wedge x_3)$ ; (b) representation by BDD. A dashed (solid) line represents the value 0 (1).

a component, and decompose the problem into two sub-problems: the first assumes the component has failed, and the second assumes it is functioning. Satyanarayana & Chang [18], and Wood [17] have shown that the factoring algorithms with reductions are more efficient at solving this problem than the classical path or cut enumeration methods. This was confirmed by the experimental works of Theologou & Carlier [19]. If too many factoring processes have to be applied, the computational time becomes prohibitive for large, or dense networks.

BDD play a key role in the automated synthesis, and formal verification of digital systems. They are the state-of-the-art data structures for representing switching functions in various branches of electronic design automation. BDD combine two advantages: the data structure is reasonably space efficient, and the algorithms that operate on them are generally time efficient. Akers [2] first introduced BDD to represent Boolean functions. Bryant popularized the use of BDD by introducing a set of algorithms for an efficient construction & manipulation of the BDD structure [3]. Their use in the reliability analysis framework has been introduced by Madre & Coudert [9], [8], and developed by Odeh [26], and Rauzy [7], [29]. They are specially used to assess fault trees in system analysis. In the network reliability framework, Sekine & Imai [13], and Trivedi [25] have shown how to functionally construct the corresponding BDD. We have improved these techniques by using the concept of partitions of network nodes. We present an exact algorithm for computing the  $K$ -terminal reliability of graph  $G = (V, E)$  with perfect vertices in  $O(|E| \cdot F_{max} \cdot 2^{F_{max}} \cdot B_{F_{max}})$ , where  $B_{F_{max}}$  is the Bell number of the maximum boundary set  $F_{max}$ .  $F_{max}$  is a network topology characteristic closely related to the linear-width. Several experiments performed on literature instances [21] show the efficiency of our method. Our method gives better results than the method proposed by Kuo, Lu, and Yeh [21]. In addition, the reliability of some large networks (up to hundreds of nodes with small  $F_{max}$ ) has been computed.

This paper is organized as follows. First, we illustrate the preliminaries of BDD in Section II. In Section III, we define the network reliability problems. Then the algorithm *BDDPart()* for constructing the BDD of a  $K$ -terminal network is shown in Section IV. Next, in Section V, experimental results on several networks are shown. Finally, we draw some conclusions in Section VI.

## II. BINARY DECISION DIAGRAM (BDD)

In this section, we recall basics about BDD. For a more detailed presentation, the reader should see the [2], [3], and [6].

### A. Representation

The BDD structure provides compact representations of Boolean expressions. The BDD associated with a Boolean expression is a compact encoding of the truth table of this expression. A BDD is a DAG based on the Shannon decomposition. The Shannon decomposition for a Boolean function  $f$  is defined as follows:

$$f = x f_{x=1} + \bar{x} f_{x=0}$$

where  $x$  is one of the decision variables, and  $f_{x=i}$  is the Boolean function  $f$  evaluated at  $x = i$ . By choosing a total order over the variables, and applying recursively the Shannon decomposition, the truth table of any Boolean expression can be graphically represented as a binary tree. Sink nodes are labeled either with 0, or with 1, representing the two corresponding constant expressions. Each internal node  $u$  is labeled with a Boolean variable  $var(u)$ , and has two out-edges called 0-edge, and 1-edge. The node linked by the 1-edge represents the Boolean expression when  $x_i = 1$ , i.e.  $f_{x_i=1}$ ; while the node linked by the 0-edge represents the Boolean expression when  $x_i = 0$ , i.e.  $f_{x_i=0}$ . The two outgoing edges are given by two functions  $low(u)$ , and  $high(u)$ . Fig. 1(a) shows the Shannon tree of Boolean function  $(x_1 \wedge x_3) \vee (x_2 \wedge x_3)$ .

Indeed, such representation is space consuming. It is however possible to shrink it by means of the following three transformation rules:

- (i) Eliminate all but one terminal vertex with a given label, and redirect all arcs into the eliminated vertices to the remaining one.
- (ii) If non-terminal vertices  $u$ , and  $v$  have  $var(u) = var(v)$ ,  $low(u) = low(v)$ , and  $high(u) = high(v)$ , then eliminate one of the two vertices, and redirect all incoming arcs to the other vertex.
- (iii) If non-terminal vertex  $v$  has  $low(v) = high(v)$ , then eliminate  $v$ , and redirect all incoming arcs to  $low(v)$ .

The transformation rules must be applied repeatedly because each transformation can expose new possibilities for further

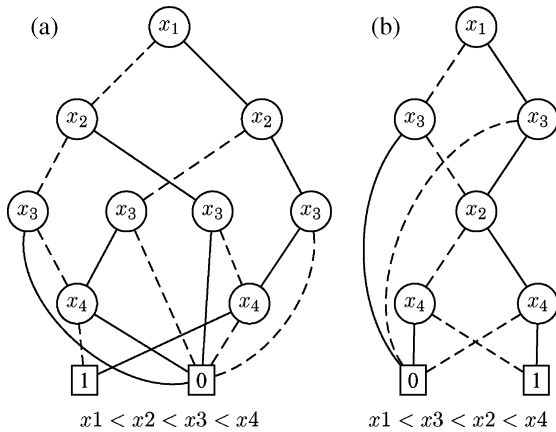


Fig. 2. BDD of Boolean function  $(x_1 \leftrightarrow x_3) \wedge (x_2 \leftrightarrow x_4)$  with two different variable orderings.

ones. The BDD associated with the Boolean expression is then constructed. The BDD representation of a function is canonical for a given edge ordering. Fig. 1(b) illustrates the BDD representation of the Boolean function  $(x_1 \wedge x_3) \vee (x_2 \wedge x_3)$  after reduction of its decision tree.

### B. Effect of Variable Ordering

The size of a BDD (i.e. the number of nodes), and hence the complexity of its manipulation, heavily depends on the variable ordering. In many applications, BDD with a huge size can be generated, which can render a BDD-based analysis impractical or inefficient. Determining the optimal variable ordering is also an NP-complete problem [22]. Basically, for a function of  $n$  variables, there are  $n!$  different variable orderings. The best known algorithm to obtain the optimal variable ordering has a complexity of  $O(3^n)$  [22]. Fig. 2 shows the effect of the variable ordering on the BDD size. Bryant shows [3] that the BDD encoding the function  $f(x_1, \dots, x_n, y_1, \dots, y_n) = (x_1 \wedge y_1) \otimes \dots \otimes (x_n \wedge y_n)$  has a linear size for the ordering  $x_1 < y_1 < \dots < x_n < y_n$ , and an exponential size for the ordering  $x_1 < \dots < x_n < y_1 < \dots < y_n$ . For instance, the BDD of expression  $(x_1 \leftrightarrow x_3) \wedge (x_2 \leftrightarrow x_4)$  with the ordering  $x_1 < x_2 < x_3 < x_4$ , consists of 11 nodes, and 8 nodes with the ordering  $x_1 < x_3 < x_2 < x_4$  (Fig. 2).

## III. NETWORK RELIABILITY

### A. Network Connectivity Functions

Network reliability is the probability that the network can guarantee a given function for a specified period of time. This paper is concerned with the network connectivity function. Three kinds of connectivity may be distinguished:

- *All-terminal reliability*,
- *2-terminal reliability* (or *terminal-pair reliability*), and
- *K-terminal reliability*

The difference between them is the subset of sites that have to be connected to ensure the connectivity function. We define a subset of nodes  $K$  ( $K \subseteq V$ ) which are essential to the system function, and have to communicate with each other; i.e.,

the network is up iff there is at least one path made of functioning edges linking nodes in  $K$ . When the connectivity function requires that each pair of sites should be able to communicate via at least one operational path, this is the *all-terminal reliability* ( $K = V$ ). The *2-terminal reliability* is the probability that two specified sites,  $s$ , and  $t$ , can communicate together ( $K = \{s, t\}$ ). When all the sites of a subset  $K$  must be connected, it consists in evaluating the *K-terminal reliability*. The *all-terminal*, and *2-terminal reliabilities* are particular cases of the *K-terminal reliability*, but for more easiness, specific methods are frequently used to solve them. The algorithm presented in Section IV can assess these three kinds of network reliability.

### B. Network Model

Our network model is an undirected stochastic graph  $G = (V, E)$ , with  $V$  as its vertex set (representing sites), and  $E \subseteq V \times V$  as its edge set (representing the links between the sites). Each edge  $e_i$  of the stochastic graph  $G$  is subject to failure with known probability  $q_i$  ( $q_i \in [0, 1]$ ). We denote  $p_i = 1 - q_i$  as the probability that edge  $e_i$  functions, and assume that all the failure events are statistically independent. In the following, we consider the vertices as perfect. In classical enumerative methods, all the states of the graph are generated, evaluated as a failing state or a functioning state, and then probabilistic methods are used to compute the resulting reliability. So, as there are two states for each edge, there are  $2^m$  possible states for the stochastic graph  $G$ . Let  $X_i$  be the binary random variable "state of the link  $e_i$  in  $G$ ", defined by  $X_i = 1$  if link  $e_i$  is operational, and  $X_i = 0$  if link  $e_i$  is down.  $X = (X_1, \dots, X_m)$  is the *random network state vector*. A state  $x$  of  $G$  is denoted by  $x = (x_1, x_2, \dots, x_m)$ , where  $x_i$  stands for the state of edge  $e_i$ , i.e.,  $x_i = 0$  when edge  $e_i$  fails, and  $x_i = 1$  when it functions. The associated probability of  $x$  is defined as

$$Pr(X = x) = \prod_{i=1}^m (x_i \cdot p_i + (1 - x_i) \cdot q_i).$$

Each state  $x$  is associated with a partial graph  $G(x) = (V, E')$  with  $E' = \{e_i \in E : x_i = 1\}$ . For the  $K$ -terminal reliability problem,  $x$  is a functioning state if all the vertices of  $K$  belong to the same connected component in  $G(x)$ ; otherwise, it is a failing state. In Fig. 3,  $x^{(1)}$  is a functioning state because all terminal vertices are connected, and  $x^{(2)}$  is a failing one because the terminal vertex  $d$  ( $K = \{a, c, d\}$ ) is disconnected. The  $K$ -terminal reliability, denoted by  $R_K(p; G)$  (where  $p = (p_1, \dots, p_m)$ ), can be defined as:

$$R_K(p; G) = \sum_{x \text{ is a functioning state}} Pr(X = x)$$

Because of the exponential number of possible states, a basic enumerative method cannot be implemented for large networks. The complexity of such a method is  $O(m \cdot 2^m)$ , so only the reliability of small networks can be computed. During such state enumeration process, some substates are equivalent, so there are numerous redundancies in the calculation.

We now define two graph operations: the *edge deletion*, and the *edge contraction*. Let  $G = (V, E)$  be a given graph such

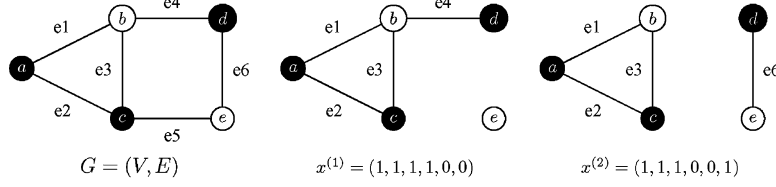


Fig. 3. The black nodes represent terminal nodes.  $x^{(1)}$  (resp.  $x^{(2)}$ ) represents a functioning (resp. failing) state.

that there is an edge  $e_i \in E$ . Then, we denote  $G_{-i}$  to be the subgraph obtained from  $G$  by deleting  $e_i$  ( $G_{-i} = G \setminus \{e_i\}$ ). If  $e_i = (x, y)$  ( $x$ , and  $y \in V$ ), the edge contraction consists in merging vertices  $x$ , and  $y$  in one single vertex. We denote  $G_{*i}$  to be the subgraph obtained from  $G$  by contracting  $e_i$ . When edge  $e_i$  fails, the network behavior is equivalent to  $G_{-i}$ ; and when  $e_i$  functions, the network behavior is equivalent to  $G_{*i}$ . Based on this decomposition, the following well-known result will be used in the sequel:

$$R_K(p; G) = p_i \cdot R_K(p; G_{*i}) + q_i \cdot R_K(p; G_{-i}).$$

These operations can be recursively applied: let  $e_i$ , and  $e_j$  be two edges of  $E$ ;  $G_{*i*j}$  is the subgraph achieved by contracting  $e_i$ , and  $e_j$ ; and  $G_{*i-j}$  is the subgraph achieved by contracting  $e_i$ , and by deleting  $e_j$ . These graph operations will be used in the following part.

#### IV. ENCODING, AND EVALUATING THE NETWORK RELIABILITY BY BDDPART()

This section shows how to use the BDD to compute the network reliability. The method complexity is thus given.

##### A. The $K$ -Terminal Reliability Boolean Function

The BDD is a compact representation of the entire set of the functioning, and failing network states. It keeps off the enumeration redundancies by merging equivalent substates. The  $K$ -terminal network function is represented by the Boolean function  $f : \{0, 1\}^m \rightarrow \{0, 1\}$  defined as:

$$\begin{cases} f(x_1, x_2, \dots, x_m) = 1 & \text{if } K \text{ - nodes are linked, and} \\ f(x_1, x_2, \dots, x_m) = 0 & \text{otherwise} \end{cases}$$

Our aim is to encode this switching function by BDD. A useful property of BDD is that all the paths from the root to the leaves are disjoint, so from this, the probability of the  $K$ -terminal connectivity is easy to compute. If  $f$  represents the system structure function, based on this property, the  $K$ -terminal reliability  $R_K$  of the network can be recursively evaluated by

$$\begin{aligned} \forall i \in [1 \dots m]: \\ R_K(p; G) &= Pr(f = 1) \\ R_K(p; G) &= p_i \cdot Pr(f_{X_i=1} = 1) + q_i \cdot Pr(f_{X_i=0} = 1) \end{aligned}$$

Recursively, we can obtain the values of  $f_{X_i=0}$ , and  $f_{X_i=1}$  until reaching the sink node (i.e.  $f_{X_i=j} = 1$ , or  $f_{X_i=j} = 0$  with  $j \in \{0, 1\}$ ). We store the probability associated with each internal node during the recursive process, so the complexity of the BDD-based reliability computation strongly depends on the size of the BDD. Algorithm *Rel\_Network()* (Fig. 4) computes

---

##### Algorithm *Rel\_Network*( $\Phi$ )

---

```

input:
 $\Phi$ : BDD /*  $\Phi = x_i \cdot \Phi_1 + \bar{x}_i \cdot \Phi_0$  */
/*  $high(\Phi) == \Phi_1$  and  $low(\Phi) == \Phi_0$  */
output:
 $R_K(p; G)$ 

if ( $\Phi == sink\_node\_1$ ) then
    return 1
end if
if ( $\Phi == sink\_node\_0$ ) then
    return 0
end if
if (entry { $\Phi, prob$ } exists in hash table) then
    return prob
else
     $P_1 = Rel\_Network(high(\Phi))$ 
     $P_2 = Rel\_Network(low(\Phi))$ 
     $prob = p_i * P_1 + q_i * P_2$ 
    /*  $p_i = Pr(x_i = 1) = 1 - q_i$  */
end if
insert entry { $\Phi, prob$ } in hash table.

return prob /* =  $R_K(p; G)$  */
    
```

---

Fig. 4. Algorithm for evaluating probability, based on BDD.

the network reliability based on the BDD  $\Phi$ , representing the network structure function. By applying this algorithm on BDD in Fig. 7(b), we obtain the  $K$ -terminal reliability

$$\begin{aligned} R_K(p; G) &= p_1 (p_2 (p_4 + q_4 p_5 p_6) \\ &\quad + q_2 (p_3 (p_4 + q_4 p_5 p_6) + q_3 p_4 p_5 p_6)) \\ &\quad + q_1 p_2 (p_3 (p_4 + q_4 p_5 p_6) + q_3 p_5 p_6) \end{aligned}$$

$R_k(p; G)$  can then be computed for all  $p$  (Fig. 7(c)).

##### B. Constructing BDD Based on Edge Contraction/Deletion

In this section, we present our method for encoding the  $K$ -terminal reliability function by BDD.

1) *Network Decomposition Based on Edge Contraction/Deletion*: The decomposition process can be represented as a binary tree so that the root corresponds to the original graph  $G$ ; and children correspond to subgraphs obtained by successive edge deletions, and edge contractions. Fig. 5 illustrates this decomposition process of network  $G$ :

- At the root (level 1), we consider edge  $e_1$ ; and we construct two subgraphs  $G_{-1}$ , and  $G_{*1}$  from the original graph  $G$ .
- Then, as a second step, we consider edge  $e_2$ . From subgraph  $G_{-1}$ , we construct subgraphs  $G_{-1-2}$ , and  $G_{-1*2}$ ;



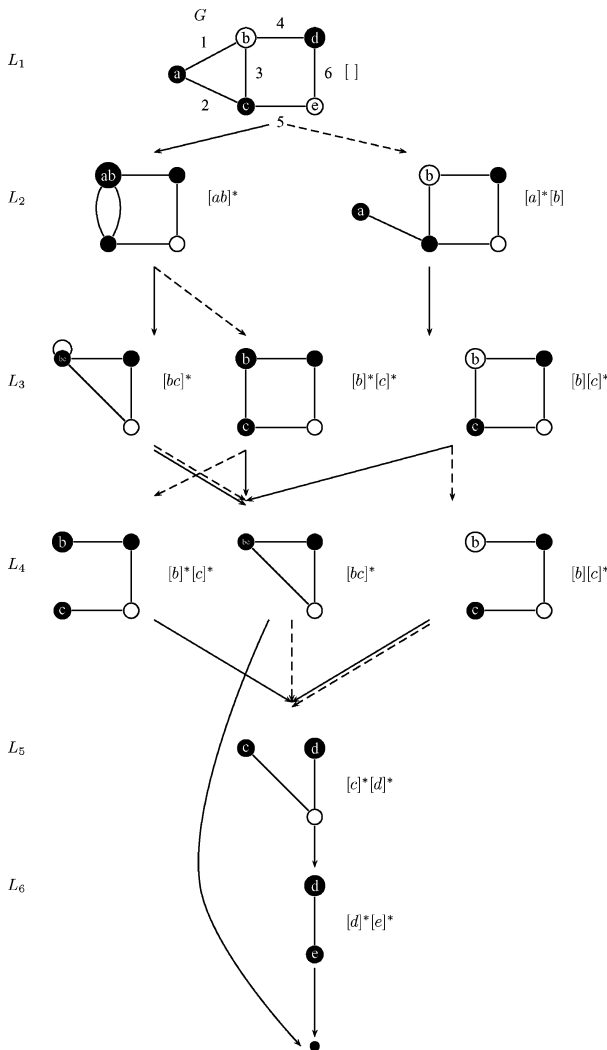


Fig. 5. Network decomposition based on edge deletion/contraction (only functioning subgraphs are represented). The equivalence between subgraph, and partition is shown.

and from subgraph  $G_{*1}$ , we construct the two corresponding subgraphs  $G_{*1-2}$ , and  $G_{*1*2}$ . The subgraph  $G_{-1-2}$  is discarded because the  $K$ -vertex  $a$  is disconnected.

- From the  $(m - 2)$  other levels, we apply the same decomposition until the vertices of  $K$  are fully connected, or at least one of them is disconnected.

The  $K$ -vertices are assuredly connected when they have been merged into one single vertex in a corresponding subgraph. All such subgraphs represent a functioning subgraph. Inversely, all subgraphs in which at least one terminal vertex is disconnected from the others represent a failing subgraph.

Obviously, isomorphic subgraphs appear in this decomposition. For instance, in Fig. 5, subgraphs  $G_{*1-2*3}$ , and  $G_{-1*2*3}$  are isomorphic (level 3). By merging these isomorphic subgraphs, redundant computations are avoided in the following steps. In this way, we have to provide an efficient method to recognize isomorphic subgraphs. We use the method introduced by Carlier & Lucet [20] for representing subgraph topologies

by the partitions of some vertices. By sharing isomorphic subgraphs, the expansion tree is modified as a rooted acyclic graph (therefore a BDD).

2) *Sharing Isomorphic Subgraphs*: Here, we present a method for efficiently recognizing isomorphic graphs during the construction of the BDD. We suppose that the initial edge ordering is  $e_1 < e_2 < \dots < e_k < \dots < e_m$ . We define  $\bar{E}_k = \{e_1, \dots, e_{k-1}\}$ , and  $\bar{E}_k = \{e_k, \dots, e_m\}$ . At level  $k$  in the BDD, each edge in  $\bar{E}_k$  has a fixed state, so is either deleted, or contracted. Consequently, the corresponding subgraphs are only made of the edge set  $\bar{E}_k$ .

*Definition 1 (Boundary Set)*: Let  $F_k$  be a vertex set such that each vertex of  $F_k$  is incident to at least one edge in  $\bar{E}_k$ , and one edge in  $\bar{E}_k$ .  $F_k$  is named *the boundary set* in level  $k$ . We denote  $F_{max}$  to be the maximum size of the boundary sets during the decomposition ( $F_{max} = \max\{|F_i|, i = 1, 2, \dots, m\}$ )

The subgraphs will be represented by the partitions of  $F_k$ .

*Definition 2 (Partition)*: To represent a subgraph by a partition in level  $k$ , we gather vertices of  $F_k$  in *blocks* according to the following rules:

- Two vertices  $x$ , and  $y$  of  $F_k$  are in the same block iff  $x$ , and  $y$  have been merged into one single vertex during the contraction/deletion process of edges in  $\bar{E}_k$ .
- A block is *marked* by an asterisk if at least one  $K$ -vertex has been merged with one of the vertices of this block, that is to say, there is a path made of up-edges ( $\in \bar{E}_k$ ) connecting a  $K$ -vertex, and at least one vertex in this block (this partition is said to be *marked*).

*Definition 3 (Isomorphic Graphs)*: Let  $G_1$ , and  $G_2$  be two subgraphs of  $G$  obtained by successive deletions, and contractions with the same edge set  $\bar{E}_k$ . The subgraphs  $G_1$ , and  $G_2$  are called *isomorphic* with the same edge ordering if their associated partitions are identical [13].

For instance, in Fig. 5, in level 4 (with boundary set  $F_4 = \{b, c\}$ ), the isomorphic subgraphs  $G_{*1*2-3}$ , and  $G_{*1-2*3}$  are represented by partition  $[bc]^*$ .

3) *Partition Classification*: Now, we order partitions in the same level  $k$  to identify & store them in an efficient way. The partition classification (i.e. each partition is associated a number) will be use during the BDD construction by *BDDPart()* for merging isomorphic nodes. Each internal node is labeled by a partition number. The order of the partitions stems from the *Stirling numbers of the second kind*, and more particularly from the following recursive formula:

$$A_{i,j} = j \cdot A_{i-1,j} + A_{i-1,j-1} \quad \text{if } 1 < j \leq i$$

with  $A_{i,1} = 1$ , and  $A_{i,j} = 0$  if  $0 < i < j$ .

$A_{i,j}$  is the number of partitions of  $j$  blocks that can be made with  $i$  elements. For a given number of elements  $i$ , the order first follows the growing number of blocks, and then uses the recursive generation; we order the partitions stemmed from the partitions of  $i - 1$  elements, and  $j$  blocks (by adding the  $i^{th}$  element in each block), then we order the partitions stemmed from partitions of  $i - 1$  elements, and  $j - 1$  blocks (by adding an  $i^{th}$  element in a new block), and so on. Each partition of the boundary set  $F_k$  has an associated number between 1, and  $B_{|F_k|}$ .

1 - [123]	12 - [12]*[3]
2 - [123]*	13 - [12][3]*
3 - [13][2]	14 - [12]*[3]*
4 - [13]*[2]	15 - [1][2][3]
5 - [13][2]*	16 - [1]*[2][3]
6 - [13]*[2]*	17 - [1][2]*[3]
7 - [1][23]	18 - [1]*[2]*[3]
8 - [1]*[23]	19 - [1][2][3]*
9 - [1][23]*	20 - [1]*[2][3]*
10 - [1]*[23]*	21 - [1][2]*[3]*
11 - [12][3]	22 - [1]*[2]*[3]*

Fig. 6. Marked partitions of  $F_k = \{1, 2, 3\}$  ( $W_{th}(|F_k|) = 22$ ), and their associated numbers ( $|K| \geq 3$ ).

The number  $B_{|F_k|}$  of such partitions is also known as the *Bell number*, and is defined as

$$B_{|F_k|} = \sum_{j=1}^{|F_k|} A_{|F_k|,j}.$$

$W$  represents the *width* of the BDD. If  $W_i$  is the number of nodes in the *ith* level, then  $W$  is defined as

$$W = \max_{i=1, \dots, n} W_i$$

*Proposition 1:* The theoretical width  $W_{th}$ , i.e. the theoretical maximum number of marked partitions, for the boundary set  $F$  is given by

$$W_{th}(|F|) = \sum_{j=1}^{|F|} \left( A_{|F|,j} \cdot \sum_{l=0}^{\min(|K|,j)} \binom{j}{l} \right). \quad (1)$$

*Proof:* There are  $\sum_{l=1}^r \binom{j}{l}$  possibilities to mark at most  $r$  blocks of a partition of  $j$  blocks ( $r \leq j$ ). A marked block corresponds to at least one  $K$ -vertex, so a partition can have as many marked blocks as the number of  $K$ -vertices. In other words, a partition of  $j$  blocks has at most  $\min(|K|, j)$  marked blocks. Hence  $A_{|F|,j} \cdot \sum_{l=0}^{\min(|K|,j)} \binom{j}{l}$  represents the number of marked partitions of  $j$  blocks. Fig. 6 shows the 22 partitions of a set of three elements. ■

The algorithm *part\_to\_number()* (Fig. 9) returns the number corresponding to a given partition, and the algorithm *number\_to\_part()* (Fig. 8) generates the partition corresponding to a given number. Both have a linear complexity, according to the number of elements.

4) *BDDPart() Algorithm:* From now on, each subgraph in the decomposition process can be represented by a marked partition, that is itself represented by a number (Fig. 6). So, we only manipulate marked partitions, and numbers instead of graphs during the BDD construction for the  $K$ -terminal reliability problem. In this way, as each BDD node is labeled by a partition number, the BDD is constructed by merging isomorphic nodes thanks to the representation of graphs by marked partitions (Fig. 7). We use a hash table to record the partition number of each shared isomorphic subgraph. When constructing partitions of level  $k$  (from

partitions in level  $k - 1$ ), we check in the hash table for an existing partition number which corresponds to the newly generated partitions. The isomorphic subgraphs are identified, and then the redundant computations are avoided. The algorithm *BDDPart()* (Fig. 10) constructs the BDD representing the  $K$ -terminal reliability function. We assume that edges are ordered by using a heuristic, based on the breadth-first-search exploration of the graph. We emphasize that the variable ordering is very important for BDD generation (see Section II-B). Time, and space complexity of BDD closely depend on this ordering. We tried several heuristics, using depth-first-search, and breadth-first-search, but the latter appears to be the best one for the great majority of the tested networks [21]. Fig. 7(a) illustrates this decomposition process of network  $G$  with the use of partition:

- **level 1** We consider edge  $e_1 = (a, b)$ . The boundary set  $F_1$  is equal to  $\{a, b\}$ . Starting for the empty partition, we construct partitions  $[ab]^*$  ( $e_1$  functions, so vertices  $a$ , and  $b$  are in the same block; and this block is marked because  $a$  is a  $K$ -vertex), and  $[a]^*[b]$  ( $e_1$  fails, so  $K$ -vertex  $a$  is alone in a marked block).
- **level 2** Vertex  $c$  is now in the new boundary set  $F_2$  (we consider edge  $e_2 = (a, c)$ ), and vertex  $a$  is not present (the two edges  $e_1$ , and  $e_2$  adjacent to  $a$  have a fixed state). So, the new boundary set  $F_2$  is equal to  $\{b, c\}$ . The partitions  $[bc]^*$ , and  $[b]^*[c]^*$  are generated from  $[ab]^*$ . Only partition  $[b][c]^*$  is generated from  $[a]^*[b]$  because the other partition corresponds to a failing state (vertex  $a$  is disconnected).
- **level 3** The boundary remains unchanged. From the three partitions in this level, we generate partitions  $[b]^*[c]^*$ ,  $[bc]^*$ , and  $[bc]^*$ . As the number of each partitions is stored in a hash table, we can detect by a hit in this table that partition  $[bc]^*$  can be shared.
- **level 4** The boundary set is now equal to  $\{c, d\}$ . By the same process, one partition is generated. One partition (from  $[bc]^*$ ) corresponds to a functioning subgraph, so we branch to leaf 1.
- **level 5** From  $[c]^*[d]^*$ , we construct a partition corresponding to a functioning subgraph (so we branch to leaf 1), and  $[d]^*[e]^*$ .
- **level 6** From  $[d]^*[e]^*$ , we branch to leaf 1 (resp. leaf 0) if  $e_6$  functions (resp. fails).

Eventually, the BDD representing the  $K$ -terminal network  $G$  is constructed (Fig. 7(b)).

### C. BDD Computation Complexity

*Theorem 1:* The  $K$ -terminal reliability computation (algorithm *BDDPart()* followed by *Rel\_Network()*) runs in  $O(m \cdot F_{max} \cdot 2^{F_{max}} \cdot B_{F_{max}})$ .

*Proof:* The BDD computation time, and its required memory depend on the number of generated & enumerated partitions at each level. As  $F_{max}$  denotes the maximum size of the boundary sets, this number of partitions is bounded by  $W_{th}(F_{max})$ . As the BDD depth is  $m$ , the total number of partitions is bounded by  $m \cdot W_{th}(F_{max})$ . From Proposition 1, we have the inequality

$$W_{th}(|F_k|) \leq \sum_{j=1}^{|F|} \left( A_{|F|,j} \cdot \sum_{l=0}^j \binom{j}{l} \right) \leq 2^{|F_k|} \cdot B_{|F_k|}$$

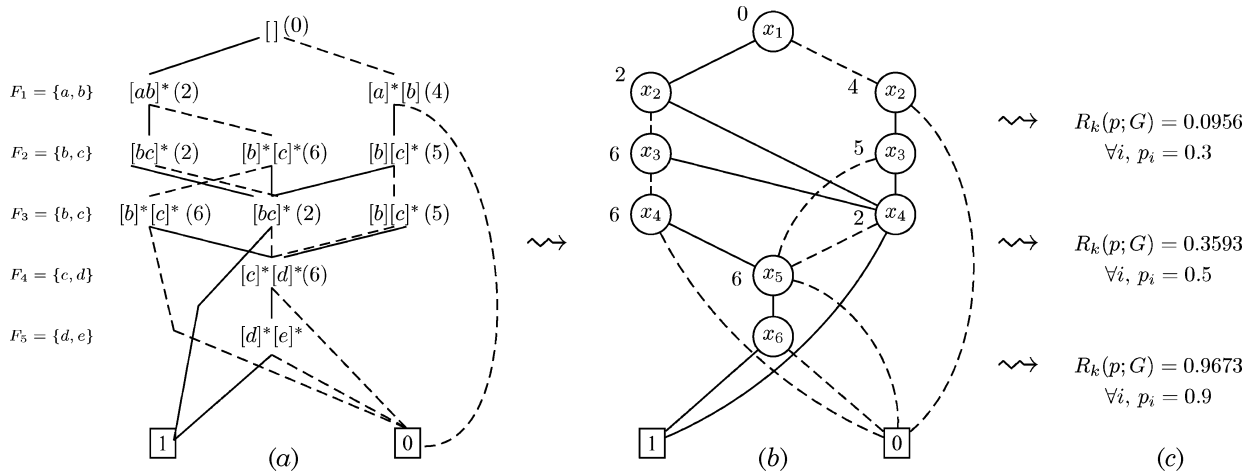


Fig. 7. The network is shown in Fig. 3. (a) BDD computation process; (b) resulting BDD; (c) various computations of  $R_k(p; G)$ .

---

**number\_to\_part(num, size)**

---

*input:*

*num*: integer (partition number)

*size*: integer (number of elements)

*output:*

partition  $part[]$  of *size* elements with number *num*

*i, j, nij*: integer

*part[]*: partition

*i = size*

find *j* such that  $\sum_{k=1}^{j-1} A_{i,k} < num < \sum_{k=1}^j A_{i,k}$

$nij = num - \sum_{k=1}^{j-1} A_{i,k}$

**while** ( $i \neq 1$ ) **do**

**if** ( $nij \leq j * A_{i-1,j}$ ) **then**

$part[i] = nij - floor((nij - 1)/j) * j$

$nij = floor((nij - 1)/j) + 1$

**else**

$nij = nij - j * A_{i-1,j}$

$part[i] = j$

$j = j - 1$

**end if**

$i = i - 1$

**end while**

$part[1] = 1$

**return**  $part[]$

---

Fig. 8. Algorithm for constructing a partition of *size* elements from a number *num*.

At level  $k$ , at most  $W_{th}(|F_k|)$  partitions are generated ( $number\_to\_part()$ ) in  $\theta(|F_k|)$ . For each of them, we compute the two corresponding partitions of level  $k + 1$ , first by deleting  $e_{k+1}$  in  $\theta(|F_k|)$ , and next by contracting  $e_{k+1}$  in  $\theta(|F_k|)$ . These partitions are encoded by a number in  $\theta(|F_k|)$  ( $part\_to\_number()$ ). Consequently, the algorithm time complexity is bounded by  $O(m \cdot F_{max} \cdot 2^{F_{max}} \cdot B_{F_{max}})$ , and the required memory complexity is bounded by  $O(m \cdot 2^{F_{max}} \cdot B_{F_{max}})$ . ■

*Corollary 1:* The 2-terminal reliability computation ( $BD-Part()$ ) followed by  $Rel\_Network()$  runs in  $O(m \cdot F_{max}^3 \cdot B_{F_{max}})$ .

---

**part\_to\_number(part[])**

---

*input:*

*part[]*: partition

*output:*

number *num* of partition  $part[]$

$j = 1, l, num = 1$ : integer

**for all** ( $l$  such that  $2 \leq l \leq |F_i|$ ) **do**

**if** ( $part[l] = j + 1$ ) **then**

$j = j + 1$

$num = num + j * A_{l-1,j}$

**else**

$num = j * (num - 1) + part[l]$

**end if**

**end for**

**for all** ( $l$  such that  $1 \leq l \leq j - 1$ ) **do**

$num = num + A_{F_i,l}$

**end for**

**return** *num*

---

Fig. 9. Algorithm for computing the number of partition  $part[]$ .

*Proof:* In this case, from formula 1,

$$W_{th}(|F|) = \sum_{j=1}^{|F|} \left( A_{|F|,j} \cdot \sum_{l=0}^{\min(2,j)} \binom{j}{l} \right)$$

$$W_{th}(|F|) \leq \sum_{j=1}^{|F|} \left( A_{|F|,j} \cdot \sum_{l=0}^2 \binom{j}{l} \right)$$

$$W_{th}(|F|) \leq \sum_{j=1}^{|F|} (A_{|F|,j} \cdot (j^2 + 1))$$

$W_{th}(F_{max})$  is then bounded by  $(F_{max}^2 + 1) \cdot B_{F_{max}}$ , and we easily deduce that the complexity of the 2-terminal reliability computation is bounded by  $O(m \cdot F_{max}^3 \cdot B_{F_{max}})$ . ■

**BDDPart(G, K)***input:* $\overline{G} = (V, E)$  (edges are ordered from 1 to  $m$ ) $K$ : set of  $K$ -terminal vertices*output:*BDD encoding the  $K$ -terminal reliability of network  $G$ .

part[], part0[], part1[]: partitions

create node (bdd root) with number 0 in level 1

**for**  $k = 1$  to  $m$  **do**  Compute  $F_{k+1}$   **for each**  $node_i$  in  $L_k$  (level  $k$ ) **do**    /\*  $node_i$  represents a BDD node labelled by  $i$  \*/     $p = number\_to\_part(i, |F_k|)$     create  $part0$  in function of  $part$ ,  $F_{k+1}$ , and  $x_k = 0$ .    create  $part1$  in function of  $part$ ,  $F_{k+1}$ , and  $x_k = 1$ .    **if** (all  $K$ -vertices are in the same block in  $part1$ ) **then**  
      branch  $high(node_i)$  to terminal node 1    **else**       $j = part\_to\_number(part1)$       **if** ( $node_j$  in  $L_{k+1}$  is not in hash table) **then**        create  $node_j$  in  $L_{k+1}$         insert  $node_j$  in hash table      **end if**      branch  $high(node_i)$  to  $node_j$     **end if**    **if** (one  $K$ -vertex is disconnected in  $part0$ ) **then**      branch  $low(node_i)$  to terminal node 0    **else**       $j = part\_to\_number(part0)$       **if** ( $node_j$  in  $L_{k+1}$  is not in hash table) **then**        create  $node_j$  in  $L_{k+1}$         insert  $node_j$  in hash table      **else**        branch  $low(node_i)$  to  $node_j$       **end if**    **end if**  **end for****end for**

Fig. 10. Algorithm for computing the BDD encoding the  $K$ -terminal reliability.

*Corollary 2:* The all-terminal reliability computation runs in  $O(m \cdot F_{max} \cdot B_{F_{max}})$ .

*Proof:* We do not need to mark partitions (in this case:  $W_{th}(F_{max}) = B_{F_{max}}$ ). Similarly, the complexity is bounded by  $O(m \cdot F_{max} \cdot B_{F_{max}})$ . ■

Unfortunately, the width  $W_{th}(F_{max})$  grows exponentially with  $F_{max}$ , so the complexity of  $BDDPart()$  grows exponentially with the maximum size of the boundary set. The exponential factor of its complexity is  $F_{max}$ ; in other words,  $BDDPart()$  is efficient for networks that induce a small maximal boundary set, independently of the size (number of nodes, and edges). It is a distinct improvement compared to the other methods. The  $F_{max}$  value depends on the initial edge ordering, and the network topology. The initial ordering can be improved to reduce the BDD width, and then try to reach the optimal width  $W_{opt}$  induced by the network topology. This optimal width is well-known for specific types of networks:

- $W_{opt} = 1$  for tree networks,
- $W_{opt} = W_{th}(|V| - 1)$  for fully connected networks, and

- $W_{opt} = W_{th}(2)$  for series-parallel networks

Generally,  $W_{opt}$  is a function of the linear width of the network. The linear width of a network is the minimal  $F_{max}$  value over all the possible edge orderings [28].

## V. EXPERIMENTAL RESULTS

Computations are done by using a *Pentium 4* workstation with 512 MB memory. Our program is written in the *C* language, and needs the *GMP* library (portable library written in *C* for arbitrary precision arithmetic on integers) because the size of some partitions numbers may exceed 4 bytes.

The heuristic for ordering edges in the experiments, and therefore the variables in BDD, is known as a breadth-first-search (BFS) ordering. The computation speed heavily depends on  $F_{max}$ , and therefore the edge ordering. We consider the benchmark of networks collected in literature by Kuo, Lu, & Yeh (see [21] for a complete description of these networks). For each network, the  $K$ -terminal reliability for  $|K| = 2$ ,  $|K| = |V|/2$ , and  $|K| = |V|$  is computed. The experimental results are shown in Table I. The running time includes the computation of BDD ( $BDDPart()$ ), plus the  $K$ -terminal reliability computation ( $Rel\_Network()$ ). The columns *time* compare the CPU time (in seconds) obtained by our method, and by the method presented by Kuo *et al.* [21] on a SPARC 20 workstation. Our method displays better results than the method presented by Kuo *et al.* in [21]. In short, this method computes all the 2-terminal reliability functions for the  $(k - 1)$  terminal-pairs, then the  $K$ -terminal reliability function can be obtained by combining these  $(k - 1)$  reliability expressions with Boolean *AND* operations. This method becomes prohibitive when the number of network nodes, and/or  $K$  turns out to be important contrary to our method, which is  $K$ -independent, and directly handles  $K$ -terminal reliability (see network  $2 \times 100$ ). Moreover, these graphs have small linear-width ( $F_{max} < 6$ ) so our method displays excellent results.

Fig. 11 shows the uniform all-terminal reliability of fully connected networks  $K_n$  of  $n$  vertices, and  $n(n - 1)/2$  edges for  $n = 5$  to 14 (*uniform* means that all the edges have the same functioning probability  $p$ ). Table II displays characteristics of corresponding computed BDD. Obviously, this topology exhibits the best reliability. For a given reliability  $p$ , the larger the size is, the more reliable the network is. We can see that the reliability monotonically increases as the size of the fully connected graphs increases. The reliabilities of this kind of network are the hardest to compute. In this case, because  $F_{max} = |V| - 1$ , the exponential factor of our algorithm is the network size. Moreover, the memory space allows us to store BDD up to  $65.10^6$  nodes (the case of BDD representing the network reliability of  $K_{15}$ ).

Table III shows the results on  $n \times m$  grid networks. As  $F_{max}$  is equal to  $\min(n, m)$ , this kind of network can be computed by  $BDDPart()$  if  $n$  or  $m$  remains small. Grid networks of thousands of nodes with small linear-width ( $F_{max}$  up to 12) can be quickly computed. Based on the BDD computed in Table III, we can easily obtain a study of the evolution of the network reliability (Fig. 12).

TABLE I  
COMPARISON RESULTS FOR  $K$ -TERMINAL NETWORK RELIABILITIES (TIME 1: BDDPART() TIME 2: KUO *et al.*)

Network <i>type</i>	$ V $	$ E $	$ K  = 2$			$ K  =  V /2$			$ K  =  V $		
			TIME 1/2	/BDD/	rel.	TIME 1/2	/BDD/	rel.	TIME 1/2	/BDD/	rel.
com.6	6	15	0.03/0.02	154	0.99997	0.03/0.77	145	0.99996	0.04/0.82	174	0.99993
net2_6	20	30	0.35/0.08	65665	0.99439	1.50/1.03	424450	0.98468	0.20/0.95	45307	0.97350
net2_8	16	24	0.11/0.02	8452	0.99555	0.50/0.87	31659	0.98892	0.06/1.08	5961	0.97965
net.18	14	23	0.06/0.02	1810	0.98731	0.07/0.76	2639	0.98429	0.07/0.83	2567	0.96926
net.19	20	30	0.09/0.33	5370	0.99712	0.12/2.18	6995	0.98909	0.12/2.50	4813	0.96006
3×10	30	47	0.06/0.55	356	0.96447	0.40/2.50	415	0.95493	0.08/2.40	236	0.92405
3×12	36	57	0.05/2.88	440	0.96173	0.02/7.80	376	0.95108	0.04/7.73	290	0.91730
5×5	25	40	0.05/0.43	2425	0.97555	0.08/2.50	1794	0.95742	0.07/2.72	1397	0.93981
2×20	40	58	0.08/0.02	152	0.78448	0.07/1.33	989	0.76525	0.05/1.55	1541	0.74529
2×100	200	298	0.02/2.52	792	0.30429	0.05/1233	794	0.27664	0.09/1677	596	0.25107

TABLE II  
BENCHMARK ON FULLY CONNECTED NETWORKS

<i>type</i>	$ V $	$ E $	TIME	/BDD/	$W$	$F_{max}$
K08	8	28	0.03	2745	405	7
K09	9	36	0.06	10265	1265	8
K10	10	45	0.13	39856	3925	9
K11	11	55	0.52	160793	15105	10
K12	12	66	2.14	673934	652	11
K13	13	78	9.97	2932248	279981	12
K14	14	91	50.00	13227624	1191235	13
K15	15	105	490.00	61780095	5021561	14

TABLE III  
BENCHMARK ON GRID NETWORKS

<i>type</i>	$ V $	$ E $	TIME	/BDD/	$W$	$F_{max}$
40000x4	160004	280003	38.4	4399938	28	5
15000x5	75005	135004	65.56	6509661	84	6
7x7	49	84	0.15	39523	858	8
1000x7	8008	15007	85.18	6559609	858	8
8x8	64	112	0.7	179410	2860	9
300x8	2709	5108	109.6	7603720	2860	9
400x8	3208	6007	147.71	10149120	2860	9
9x9	81	144	3.16	797916	9724	10
10x10	100	180	14.75	3495491	33592	11
11x11	121	220	67.94	15137188	117572	12
16x11	176	325	101.2	33360848	117572	12
21x11	231	430	101.2	24249018	117572	12
12x12	144	264	321.3	64959137	416024	13

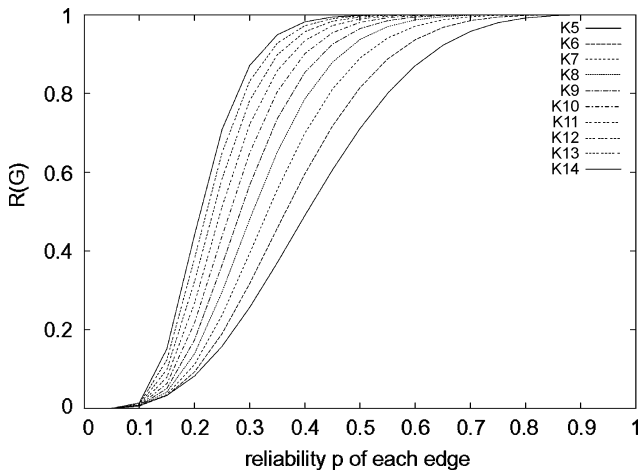


Fig. 11. Fully connected networks Evolution of the all-terminal reliability as a function of  $p$ .

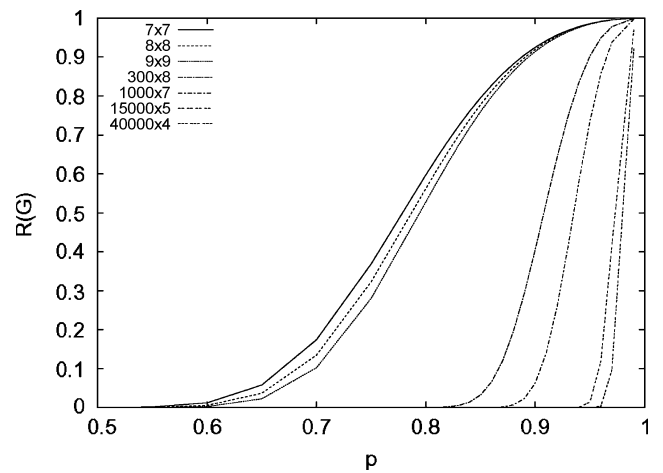


Fig. 12. Grids  $n \times m$  Evolution of the all-terminal reliability as a function of  $p$ .

## VI. CONCLUSION

A method for evaluating the  $K$ -terminal reliability via BDD has been proposed in this paper. The difficult problem of efficiently identifying the isomorphic subgraphs met during the BDD construction process has been resolved. Our algorithm runs in  $O(m \cdot F_{max} \cdot 2^{F_{max}} \cdot B_{F_{max}})$ , so the exponential factor only depends on the value of  $F_{max}$ . This algorithm was run on literature instances, and supplied good results. The reliability of networks with hundreds of nodes could be computed because  $F_{max}$  remains small.

## ACKNOWLEDGMENT

The authors would like to thank the referees for their helpful comments on this paper.

## REFERENCES

- [1] Institute of Electrical and Electronics Engineers, "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries," New York, NY, 1990.
- [2] B. Akers, "Binary decision diagrams," *IEEE Trans. Computers*, vol. C-27, pp. 509–516, Jun. 1978.
- [3] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.

- [4] M. O. Ball, "Complexity of network reliability computations," *Networks*, vol. 10, pp. 153–165, 1980.
- [5] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Trans. Reliability*, vol. R-35, no. 3, pp. 230–239, Aug. 1986.
- [6] H. Andersen, "An introduction to binary decision diagrams," Oct. 1997, Lecture Notes for 49285 Advanced Algorithm E97.
- [7] A. Rauzy, "New algorithms for fault tolerant trees analysis," *Reliability Engineering and System Safety*, vol. 5, no. 59, pp. 203–211, 1993.
- [8] O. Coudert and J. C. Madre, "Implicit and incremental computation of primes and essential primes of Boolean functions," in *Proc. of the 29th ACM/IEEE Design Automation Conference*, 1992, pp. 36–39, IEEE Computer Society Press.
- [9] O. Coudert and J. C. Madre, "A new method to compute prime and essential prime implicants of Boolean functions," in *Proc. of MIT VLSI Conference*, March 1992.
- [10] S. J. Friedman and K. J. Supowit, "Finding an optimal variable ordering for binary decision diagrams," *IEEE Trans. Computers*, vol. C-39, no. 5, pp. 710–713, 1990.
- [11] J. S. Provan, "The complexity of reliability computations on planar and acyclic graphs," *SIAM J. Computing*, vol. 15, no. 3, pp. 694–702, 1986.
- [12] G. Hardy, C. Lucet, and N. Limnios, "Computing all-terminal reliability of stochastic networks with binary decision diagrams," in *Proc. 11th International Symposium on Applied Stochastic Models and Data Analysis*, May 2005.
- [13] H. Imai, K. Sekine, and K. Imai, "Computational investigations of all-terminal network reliability via BDDs," *IEICE Transactions on Fundamentals*, vol. E82-A, no. 5, pp. 714–721, May 1999.
- [14] S. H. Ahmad, "Simple enumeration of minimal cutsets of acyclic directed graph," *IEEE Trans. Reliability*, vol. R-27, no. 5, pp. 484–487, Dec. 1988.
- [15] M. O. Locks, "A minimizing algorithm for sum of disjoint products," *IEEE Trans. Reliability*, vol. R-36, no. 4, pp. 436–445, Oct. 1987.
- [16] S. Hariri and C. S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cutset methods," *IEEE Trans. Computers*, vol. C-36, no. 10, pp. 1224–1232, Oct. 1987.
- [17] R. K. Wood, "A factoring algorithm using polygon-to-chain reductions for computing K-terminal network reliability," *Networks*, vol. 15, pp. 173–190, 1985.
- [18] A. Satyanarayana and M. K. Chang, "Network reliability and the factoring theorem," *Networks*, vol. 13, pp. 107–120, 1983.
- [19] O. Theologou and J. Carlier, "Factoring and reductions for networks with imperfect vertices," *IEEE Trans. Reliability*, vol. R-40, pp. 210–217, 1991.
- [20] J. Carlier and C. Lucet, "A decomposition algorithm for network reliability evaluation," *Discrete Applied Mathematics*, vol. 65, pp. 141–156, 1996.
- [21] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo, "OBDD-based evaluation of k-terminal network reliability," *IEEE Trans. Reliability*, vol. R-51, no. 4, 2002.
- [22] S. J. Friedman and K. J. Supowit, "Finding the optimal variable ordering for binary decision diagrams," in *Proc. 24th ACM/IEEE Conference on Design Automation*, 1987, pp. 348–356.
- [23] M. S. Choi and C. H. Jun, "Some variant of polygon-to-chain reductions in evaluating reliability of undirected network," *Microelectron. Reliab.*, vol. 35, no. 1, pp. 1–11, 1995.
- [24] J. P. Gadani, "System effectiveness evaluation using star and delta transformations," *IEEE Trans. Reliability*, vol. R-30, no. 1, pp. 43–47.
- [25] X. Zang, H. Sun, and K. Trivedi, "A BDD-based algorithm for reliability evaluation of phased mission systems," *IEEE Trans. Reliability*, vol. R-48, no. 1, pp. 50–60, 1999.
- [26] K. Odeh, "New Algorithms for Probabilistic and Logic Analysis of Fault Trees," (in French) PhD-thesis, Université de Technologie de Compiègne, French, , 1995.
- [27] C. Lucet and J.-F. Manouvrier, "Exact methods to compute network reliability," in *Statistical and Probabilistic Models in Reliability*, D. C. Ionescu and N. Limnios, Eds. Birkhauser Boston: , 1999, pp. 279–294.
- [28] H. Bodlaender and D. Thilikos, Computing Small Search Numbers in Linear Time Dept. of Computer Science, Utrecht University, Technical Report No. UU-CS-1998-05, 1998.
- [29] A. Rauzy, "A new methodology to handle Boolean models with loops," *IEEE Trans. Reliability*, vol. R-52, no. 1, pp. 96–105, 2003.

**Gary Hardy** received the BS (2002), and MS (2004) degrees in computer science from the University of Picardie Jules Verne, France. He is currently working toward the Ph.D degree in Computer Science at the University of Picardie Jules Verne. His research interests include network reliability, and graph theory.

**Corinne Lucet** received her Ph.D degree in computer science in 1993 from the Technology University of Compiègne (France). She is currently an associate professor in the Computer Science Department of Technology University of Amiens (France). She has been a member of the Laboratory of Computer Science in the University of Picardie Jules Verne since 1999. Her main research interests deal with Operations Research, and Network Optimization.

**Nikolaos Limnios** is a Professor of Applied Mathematics in the University of Technology of Compiègne (France). His main activities concern research and teaching of reliability and dependability of systems, Applied probability, and Statistics. He is the (co-) author, and co-editor of several books in reliability.



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

Computers & Operations Research 33 (2006) 2189–2207

computers &  
operations  
research

[www.elsevier.com/locate/cor](http://www.elsevier.com/locate/cor)

## An exact method for graph coloring<sup>☆</sup>

C. Lucet<sup>a</sup>, F. Mendes<sup>a,\*</sup>, A. Moukrim<sup>b</sup>

<sup>a</sup>*LaRIA FRE 2733, 5 rue du Moulin Neuf 80000 Amiens, France*

<sup>b</sup>*HeuDiaSyC UMR CNRS 6599 UTC, BP 20529 60205 Compiègne, France*

Available online 5 March 2005

---

### Abstract

We are interested in the graph coloring problem. We propose an exact method based on a linear-decomposition of the graph. The complexity of this method is exponential according to the linearwidth of the entry graph, but linear according to its number of vertices. We present some experiments performed on literature instances, among which COLOR02 library instances. Our method is useful to solve more quickly than other exact algorithms instances with small linearwidth, such as *mug* graphs. Moreover, our algorithms are the first to our knowledge to solve the COLOR02 instance *4-Inser\_3* with an exact method.

© 2005 Elsevier Ltd. All rights reserved.

*Keywords:* Graph coloring; Exact method; Linearwidth; Linear-decomposition

---

### 1. Introduction

The notions of tree-decomposition and path-decomposition have been introduced by Robertson and Seymour [1]. The decomposition method we propose here is strongly related to these notions, which have been studied in particular by Bodlaender to solve some NP-hard problems [2].

Our approach is a method based on successive decompositions of the representative graph providing successive resolved subgraphs and their corresponding separating sets named *boundary sets*. At each

---

<sup>☆</sup> With the financial support of Conseil Régional de Picardie and FSE.

\* Corresponding author.

*E-mail addresses:* [Corinne.Lucet@laria.u-picardie.fr](mailto:Corinne.Lucet@laria.u-picardie.fr) (C. Lucet), [Florence.Mendes@laria.u-picardie.fr](mailto:Florence.Mendes@laria.u-picardie.fr) (F. Mendes), [Aziz.Moukrim@hds.utc.fr](mailto:Aziz.Moukrim@hds.utc.fr) (A. Moukrim).

step, solutions of the resolved subgraph are represented by the different states of the boundary set vertices. The number of states to enumerate grows exponentially with the size of the boundary set. Its maximum size, for an optimal vertex numbering, corresponds to the linearwidth of the graph. The main advantage of this method is that the exponential factor of its complexity does not depend on the size of the graph but only on its linearwidth. This technique has been implemented efficiently by Carlier, Lucet and Manouvrier to solve various NP-hard problems such as network reliability or minimal Steiner tree computation [3–5].

We apply the decomposition method to one of the most studied problems of combinatorial optimization: the graph coloring problem. It constitutes a central problem in a lot of applications such as school timetabling, scheduling, or frequency assignment [6,7]. The graph coloring problem consists in coloring the vertices of a graph with a minimum number of colors, ensuring that two adjacent vertices do not receive the same color. Various heuristic approaches have been proposed for this NP-hard problem [8]: greedy algorithms such as DSATUR [9], metaheuristics based on local search, tabu method, simulated annealing, hybrid algorithms, etc. (see for example [10–16]). To our knowledge, few exact methods are proposed to resolve this problem. One of the most well-known exact algorithms is the exact branch-and-bound algorithm implemented by Brelaz that uses DSATUR principles [9]. Implicit enumeration strategies are used in [17–19]. Mehrotra and Trick [20] studied a linear programming formulation which is solved by using column generation techniques. More recently, Mendez Diaz and Zabala presented a branch-and-cut algorithm [21,22]. Herrmann and Hertz presented efficient algorithms used to find edge-critical and vertex-critical subgraphs that have same chromatic numbers as initial graphs but are easier to solve [23]. Desrosiers, Galinier and Hertz proposed different algorithms to detect these critical subgraphs [24]. They made experiments on random graphs and on different types of benchmark graphs. Their method is very efficient on several instances families.

Our paper is organized as follows. In Section 2, we describe the decomposition method and introduce the necessary notions. In Section 3, we develop the implementation of the method. We present an exact algorithm which enables us to solve efficiently large instances whose linearwidth is bounded. Computational results obtained on various instances are presented in Section 4. They compare with two exact methods: a branch-and-cut algorithm [22] and an algorithm based on vertex-critical subgraphs detection [24]. Finally, we give some conclusions and discuss about the perspectives of this work.

## 2. Graph decompositions

To introduce the kind of decomposition that we use to solve the graph coloring problem, it is necessary to recall some graph theory definitions and the notions of tree-decomposition and path-decomposition.

### 2.1. Preliminary definitions

An *undirected graph*  $G$  is a pair,  $G = (V, E)$ , made up of a vertex set  $V$  and an edge set  $E \subset V \times V$ . A graph  $G$  is *connected* if for all vertices  $w, v \in V (w \neq v)$ , there exists a path from  $w$  to  $v$ . Without loss of generality, the graphs  $G$  we will consider in the following of this paper will



be undirected and connected graphs. A *subgraph* of  $G = (V, E)$ , induced by  $W \subseteq V$ , is a graph  $G(W) = (W, E_W)$  such that  $E_W = E \cap (W \times W)$ . A *tree* is a simple undirected graph,  $T = (I, E_T)$ , without cycle and with  $|E_T| = |I| - 1$ . A *rooted tree* is a tree directed from the root  $r$  to the leaves. If the edge  $(p, v)$  belongs to a rooted tree,  $p$  is the *father* of  $v$ , and  $v$  is one of the *sons* of  $p$ .

## 2.2. Tree-decomposition

A *tree-decomposition* of  $G = (V, E)$  is a pair  $(\{X_i/i \in I\}, T = (I, E_T))$  with  $\{X_i/i \in I\}$  a family of subsets of  $V$  and  $T$  a tree such that:

- $\bigcup_{i \in I} X_i = V$ ,
- for all edges  $(v, w) \in E$ , there exists a subset  $X_i, i \in I$ , with  $v \in X_i$  and  $w \in X_i$ ,
- for all  $i, j, k \in I$ , if  $j$  is on the path from  $i$  to  $k$  in  $T$  then  $X_i \cap X_k \subseteq X_j$ .

The *treewidth* of a tree-decomposition is  $\max_{i \in I} (|X_i| - 1)$ . The treewidth of a graph  $G$  is the minimum treewidth over all possible tree-decompositions of  $G$ .

The decomposition method is as follows. Given a tree-decomposition of the graph, partial solutions are built on the subsets  $X_i$  and then associated to solve the considered problem. The decomposition method computes the solutions from the leaves to the root of the tree  $T$ , by examining all partial solutions on every subgraph  $G(X_i)$ . The number of partial solutions is exponential according to the size of the subgraphs  $X_i$ . These partial solutions are computed from the solutions of  $X_f$ , for all  $f$  belonging to the sons of  $i$  in  $T$ . Unlike a simple enumerative method, this method allows one to factorize partial solutions of the  $X_i$  sets into classes. This factorization provides an efficient method if the cardinality of the sets  $X_i$  is small, i.e. if the treewidth of the tree-decomposition is sufficiently small.

Whereas for some graph families, such as trees and serie-parallel graphs, one can compute the treewidth in linear time, computing the treewidth of any graph is a NP-complete problem [25]. Bodlaender [26] gives for a constant  $k$  an algorithm in  $O(n)$  which for a graph  $G$  solves the problem “is the treewidth of  $G$  at most  $k$ ?”. If so, it determines a tree-decomposition with treewidth at most  $k$ . This algorithm based on clique search and graph contraction has an exponential complexity with respect to  $k$  ( $O(n * 2^{k^2})$ ). It cannot be used in practice, even for  $k = 4$ .

## 2.3. Path-decomposition and linear-decomposition

The decomposition method that we will use in the following is based on a special case of tree-decomposition. We will consider a tree with only one leaf, that is a path.

A *path-decomposition*  $(X_1, \dots, X_r)$  of a graph  $G$  is an ordered sequence of sub-sets of  $V$  such that:

- $\bigcup_{1 \leq i \leq r} X_i = V$ ,
- for all edges  $(v, w) \in E$ , there exists a subset  $X_i, 1 \leq i \leq r$ , with  $v \in X_i$  and  $w \in X_i$ ,
- for all  $i, j, k \in \{1, \dots, r\}$ , if  $i \leq j \leq k$  then  $X_i \cap X_k \subseteq X_j$ .

The *pathwidth* of a path-decomposition is  $\max_{1 \leq i \leq r} (|X_i| - 1)$ . The pathwidth of a graph is the minimum pathwidth over all possible path-decompositions of  $G$ . A *vertex linear ordering* of  $G$  is a bijection  $\mathcal{N} : V \rightarrow \{1, \dots, |V|\}$ . For more clarity, we denote  $k$  the vertex  $\mathcal{N}^{-1}(k)$ . Let  $F_i = \{j \in V \mid \exists (j, l) \in E \ j \leq i < l\} \ \forall i \in \{1, \dots, |V|\}$ . The *linearwidth* of a vertex linear ordering  $\mathcal{N}$  is  $F_{\max}(\mathcal{N}) = \max_{i \in V} (|F_i|)$ . The linearwidth of  $G$ , written  $F_{\max}(G)$ , is the minimum linearwidth over all possible vertex linear orderings of  $G$ . The linearwidth of a graph equals its pathwidth [5].

Computing the pathwidth or the linearwidth of any graph is a NP-complete problem [25], similarly as computing the treewidth of any graph. The treewidth of a graph  $G$  is smaller or equal to its pathwidth, and as a consequence the exponential factor of a tree-decomposition is smaller than that of a path-decomposition. Nevertheless, implementing the decomposition method on a linear-decomposition is easier than using a tree-decomposition. First, from a technical point of view, several  $X_i$  partial solutions may have to be stored in memory when resolving a problem with a tree-decomposition. It involves some problems of memory storage and combination of the  $X_i$  when implementing the algorithm. Moreover, creating a linear-decomposition is easier than creating a tree-decomposition. Thus, we use a vertex linear ordering of the graph to resolve the graph coloring problem with a linear-decomposition. The resolution method is then based on a sequential insertion of the vertices, using a vertex linear ordering previously determined. This will be developed in the following section.

### 3. Application to the graph coloring problem

In this section, we propose a method which uses linear-decomposition in order to solve the graph coloring problem.

#### 3.1. Problem definition

A *coloring* of a graph  $G = (V, E)$  is an assignment of a color  $c(i) \in I$  to each vertex such that  $c(i) \neq c(j)$  for all edges  $(i, j) \in E$ .

If the cardinality of  $I$  is  $k$ , the coloring of  $G$  is called a *k-coloring*. The minimum value of  $k$  for which a *k-coloring* is possible is called the *chromatic number* of  $G$  and is denoted  $\chi(G)$ . The graph coloring problem consists in finding the chromatic number of a graph.

#### 3.2. Linear decomposition principle

Consider a graph  $G = (V, E)$ . Let  $N = |V|$  and  $M = |E|$ . The vertices of  $G$  are numbered according to a linear ordering  $\mathcal{N} : V \rightarrow \{1, \dots, N\}$ . Let  $V_i$  be subset of  $V$ , made of the vertices numbered from 1 to  $i$ . Let  $H_i = (V_i, E_i)$  be the subgraph of  $G$  induced by  $V_i$ .  $F_i$  is the *boundary set* of  $H_i$ , i.e. the subset of  $V_i$  such that  $v \in F_i$  if and only if  $\exists (v, w) \in E$  and  $v \leq i < w$  (see Fig. 1). Let  $H'_i = (V'_i, E'_i)$  be the subgraph of  $G$  induced by  $V'_i = (V \setminus V_i) \cup F_i$ . Any kind of relation between the vertices of  $H_i$  and those of  $H'_i$  depends on the vertices of  $F_i$ .

The linear decomposition is a dynamic method. During the coloring we will consider  $N$  subgraphs  $H_1, \dots, H_N$  and the  $N$  corresponding boundary sets  $F_1, \dots, F_N$ . Starting from a vertex linear ordering,

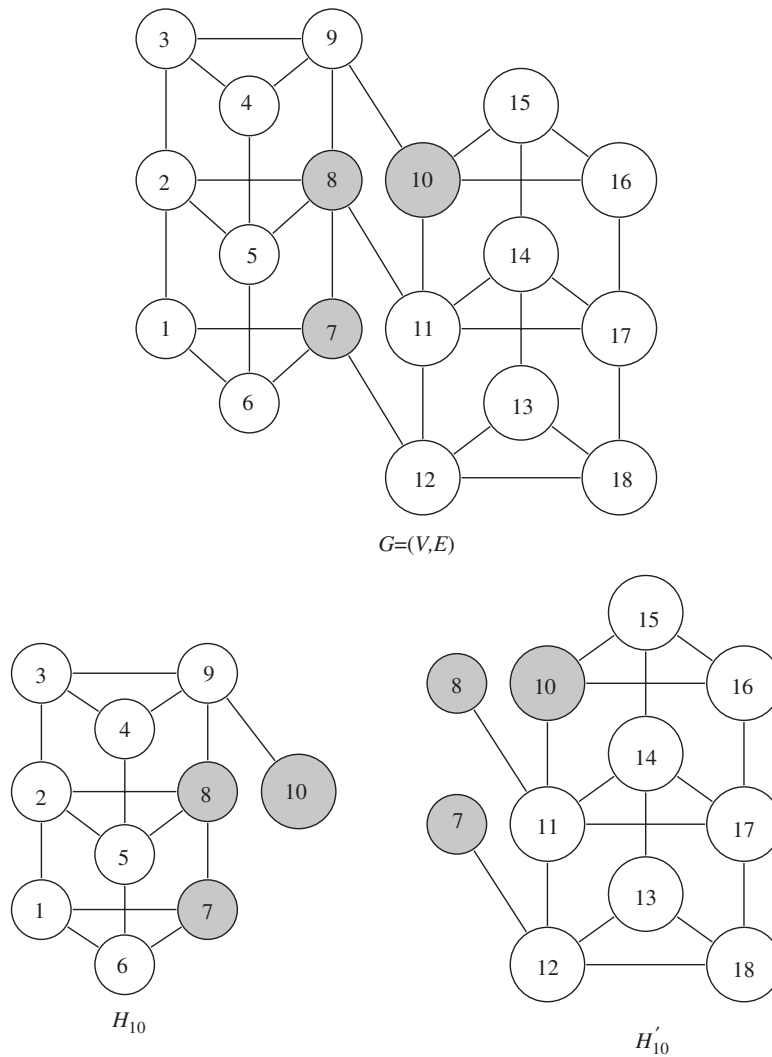


Fig. 1. Subgraph  $H_{10}$  of  $G$  and its boundary set  $F_{10} = \{7, 8, 10\}$ .

we build at first iteration a subgraph  $H_1$  which contains only the vertex 1, then at each step the next vertex and its corresponding edges are added, until  $H_N$ . Partial solutions of step  $i$  are built from partial solutions of step  $i - 1$ .

At each subgraph  $H_i$  corresponds a boundary set  $F_i$  containing the vertices of  $H_i$  which have at least one neighbor in  $H'_i$ . The boundary set  $F_i$  is built from  $F_{i-1}$  by adding the vertex  $i$  and removing the vertices that have no neighbor with an ordering number greater than  $i$ . Several colorings of  $H_i$  may correspond to the same coloring of  $F_i$  (see Fig. 2). Moreover, the colors used by the vertices  $V_i \setminus F_i$  do not interfere with the coloring of the vertices which have an ordering number greater than  $i$ , since no edge exists between them. So, only the partial solutions corresponding to different colorings of  $F_i$  have to be stored in memory. This way, several partial solutions on  $H_i$  may be summarized by a unique partial solution on  $F_i$ , called *configuration of  $F_i$* .

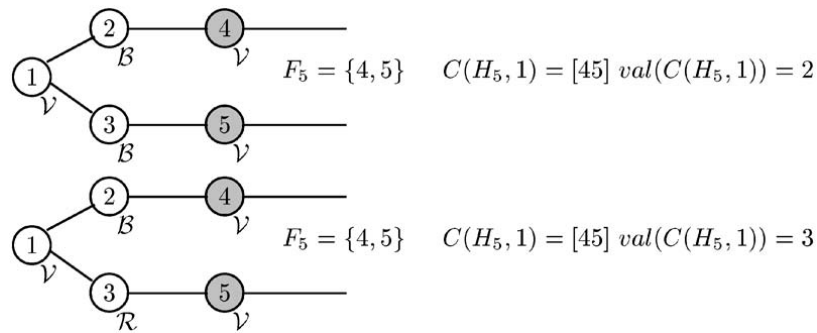


Fig. 2. Different colorings of  $H_5$  but same configuration of  $F_5$ .

The graph coloring problem is solved by evaluating at each step the configurations of the boundary set  $F_i$ . At step  $i$ , the subgraph  $H_{i-1}$  is solved. It means that to each configuration of  $F_{i-1}$  corresponds a value of the minimum number of colors necessary to color  $H_{i-1}$  for this fixed coloring of the boundary set vertices. Then, partial solutions are built using solutions of the precedent step. This point will be detailed in Section 3.4.

### 3.3. Boundary set configurations

A configuration of the boundary set  $F_i$  is a given coloring of the vertices of  $F_i$ . This can be represented by a partition of  $F_i$ , denoted  $B_1, \dots, B_j$ , such that two vertices  $u, v$  of  $F_i$  are in the same block  $B_c$  if and only if they have the same color. The number of configurations of  $F_i$  depends obviously on the number of edges between the vertices of  $F_i$ . The minimum number of configurations is 1. If the vertices of  $F_i$  form a clique, only one configuration is possible:  $B_1, \dots, B_{|F_i|}$ , with exactly one vertex in each block. The maximal number of configurations of  $F_i$  equals the number of partitions of a set with  $|F_i|$  elements. When no edge exists between the boundary set vertices, all the partitions are to be considered.

The number of partitions of a set composed by  $i$  elements and  $j$  blocks, written  $A_{i,j}$ , is given by the recursive formula of Stirling numbers of the second kind:

$$A_{i,j} = jA_{i-1,j} + A_{i-1,j-1}$$

with  $A_{1,1} = 1$  and  $A_{i,j} = 0$  if  $i < j$ .

The number  $T(F_i)$  of different partitions of the boundary set  $F_i$  equals the sum of the  $A_{|F_i|,j}$  for  $j$  from 1 to  $|F_i|$ .

$$T(F_i) = \sum_{j=1 \text{ to } |F_i|} A_{|F_i|,j}$$

To identify the configurations of the boundary set, we associate to each one an ordering number between 1 and  $T(F_i)$ . The partitions of sets with at most four elements and their ordering number are reported in Table 1.

Table 1  
Classification of the partitions of sets containing from 1 to 4 elements

$\mathcal{A}_{i,j}$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 1$	<b>1</b> [1]			
$i = 2$	<b>1</b> [12]	<b>2</b> [1][2]		
$i = 3$	<b>1</b> [123]	<b>2</b> [13][2] <b>3</b> [1][23] <b>4</b> [12][3]	<b>5</b> [1][2][3]	
$i = 4$	<b>1</b> [1234]	<b>2</b> [134][2] <b>3</b> [13][24] <b>4</b> [14][23] <b>5</b> [1][234] <b>6</b> [124][3] <b>7</b> [12][34] <b>8</b> [123][4]	<b>9</b> [14][2][3] <b>10</b> [1][24][3] <b>11</b> [1][2][34] <b>12</b> [13][2][4] <b>13</b> [1][23][4] <b>14</b> [12][3][4]	<b>15</b> [1][2][3][4]

Let  $C(H_i, x)$  be the  $x$ th configuration of  $F_i$  for the subgraph  $H_i$ . Its value, denoted  $val(C(H_i, x))$ , equals the minimum number of colors necessary to color  $H_i$  for this configuration.

In Fig. 2, two different colorings of  $H_5$  correspond to the same configuration of  $F_5$ . Only 2 colors are necessary to color  $H_5$  with the configuration for which vertices 2 and 3 have a same color, whereas 3 colors are necessary when vertices 2 and 3 have different colors. The value of the configuration  $C(H_5, 1)$ , represented by the partition [45], is 2, because we keep only the best valuation.

### 3.4. Coloring algorithm

The details of the implementation of the decomposition method are reported in algorithm 1. Note that  $H_1 = (\{1\}, \emptyset)$  and  $F_1 = \{1\}$ . So, there is only one configuration of  $F_1$ ,  $C(H_1, 1) = [1]$ . The insertion of the vertex 2 of  $G$  in  $C(H_1, 1)$  can provide one or two configurations of  $F_2$  (only one configuration if the vertices 1 and 2 are neighbors, two configurations otherwise).

At step  $i$ , we do not examine all the possible configurations of the step  $i - 1$ , but only those which have been created at precedent step, it means those for which there is no edge between two vertices of the same block. For each configuration of  $F_{i-1}$ , we introduce the vertex  $i$  in each block successively. Each time the introduction is possible without breaking the coloring rules, the corresponding configuration of  $F_i$  is generated. We generate also the configurations obtained by adding to each configuration of  $F_{i-1}$  a new block containing the vertex  $i$ .

For a given subgraph  $H_i$ , only the configurations that are different are represented. Their ordering number  $x$ , included between 1 and  $T(F_i)$ , is computed by an algorithm according to their number of blocks and their number of elements. When different colorings of  $H_i$  correspond to the same configuration of  $F_i$ , only the best valuation is kept in  $val(C(H_i, x))$ .

At step  $N$ , only one configuration  $C(H_N, 1)$  is generated from configurations of step  $N - 1$ . It represents all the optimal coloring solutions and its value equals  $\chi(G)$ .

**Algorithm 1** Coloring algorithm**Input:** a graph  $G$ **Output:**  $\chi$  : the chromatic number of  $G$  $H_1 = (\{1\}, \emptyset)$  $F_1 = \{1\}$  $C(H_1, 1) = [1]$ **for**  $i = 2$  to  $N$  **do**    Build  $H_i$  and  $F_i$     **for** each configuration  $C(H_{i-1}, x)$  of  $F_{i-1}$  **do**        **for**  $j = 1$  to number of blocks of  $C(H_{i-1}, x)$  **do**            **if**  $i$  does not have any neighbor in the block  $j$  **then**                 $part = C(H_{i-1}, x)$                 insert  $i$  in the block  $j$  of  $part$                 generate the configuration  $C(H_i, y)$  corresponding to  $part$                 **if**  $C(H_i, y)$  already exists **then**                     $val(C(H_i, y)) = \min(val(C(H_i, y)), val(C(H_{i-1}, x)))$                 **else**                     $val(C(H_i, y)) = val(C(H_{i-1}, x))$                 **end if**            **end if**        **end for**         $part = C(H_{i-1}, x)$         add to  $part$  a new block containing  $i$          $val(part) = \max(val(C(H_{i-1}, x)), \text{number of blocks of } part)$         generate the configuration  $C(H_i, y)$  corresponding to  $part$         **if**  $C(H_i, y)$  already exists **then**             $val(C(H_i, y)) = \min(val(C(H_i, y)), val(part))$         **else**             $val(C(H_i, y)) = val(part)$         **end if**    **end for****end for** $\chi = val(C(H_N, 1))$

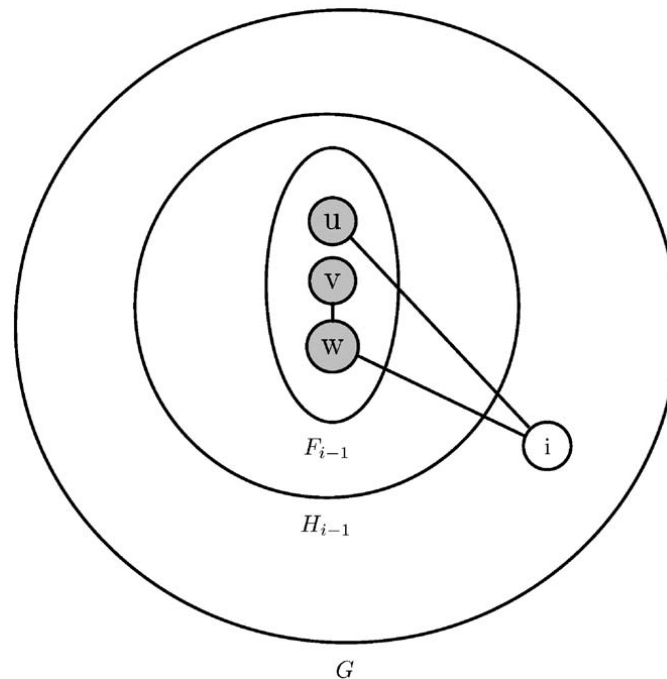


Fig. 3. Construction of  $H_i = (V_{i-1} \cup \{i\}, E_{i-1} \cup \{(u, i), (w, i)\})$ .

### 3.4.1. Example of configuration computing

Assume that we are searching for a coloring of the graph  $G$  of Fig. 3 and that we are at step  $i$  with  $F_i = \{u, v, w, i\}$ .

Suppose that at step  $i - 1$ , we had  $F_{i-1} = \{u, v, w\}$  and that the configurations of  $F_{i-1}$  were:

- $C(H_{i-1}, 2) = [uw][v]$  with value  $\alpha$ .
- $C(H_{i-1}, 4) = [uv][w]$  with value  $\beta$ .
- $C(H_{i-1}, 5) = [u][v][w]$  with value  $\gamma$ .

The values of  $\alpha$  and  $\beta$  are at least 2, since the corresponding configurations have 2 blocks. Remark that these values may be upper than 2, depending on the configurations of the preceding steps. By the same way,  $\gamma$  is at least 3.

We want to generate the configurations of  $F_i$  from the configurations of  $F_{i-1}$ .

- It is impossible to insert  $i$  in the first block of  $C(H_{i-1}, 2)$ , since  $u$  and  $i$  are neighbors. It is possible to insert  $i$  in the second block of  $C(H_{i-1}, 2)$ . We obtain  $C(H_i, 3) = [uw][vi]$  and  $val(C(H_i, 3)) = \alpha$ . It is also possible to add a third block, it provides the configuration  $C(H_i, 12) = [uw][v][i]$  with  $valC(H_i, 12) = \max(\alpha, 3)$ , since for this configuration at least three colors are needed.
- We must add a block to introduce  $i$  in the configuration  $C(H_{i-1}, 4)$ . We obtain  $C(H_i, 14) = [uv][w][i]$  with value  $\max(\beta, 3)$ .

- $i$  may be introduced in the second block of  $C(H_{i-1}, 5)$ . It provides the configuration  $C(H_i, 10) = [u][vi][w]$  with value  $\gamma$ . By adding a new block, the configuration  $C(H_i, 15)$  is created with  $val(C(H_i, 15)) = \max(\gamma, 4)$ .

Thus five configurations are provided at step  $i$ , they are used to determine the configurations of the following step, and so on until the whole graph is colored.

### 3.5. Improved coloring algorithm

The number of configurations of a boundary set  $F_i$  is exponential according to  $|F_i|$ . Moreover, a coloring represented by a configuration of  $k$  blocks needs at least  $k$  colors. By applying a succession of  $k$ -colorings (see Algorithm 2), we avoid examining the configurations of more than  $k$  blocks, which proves to be very interesting when  $F_{\max}$  is larger than  $\chi(G)$ .

Take again the example of Fig. 3, and suppose that we are now searching for a 3-coloring of the graph  $G$ . All the configurations are made of at most three blocks, so their values are upper bounded by 3. This time, only four configurations are provided at step  $i$ .

---

#### Algorithm 2 $k$ -coloring function

---

**Input:** a graph  $G$  and an integer  $k$

**Output:** *Result* : *True* if and only if  $G$  is  $k$ -colorable

$H_1 = (\{1\}, \emptyset)$

$F_1 = \{1\}$

$C(H_1, 1) = [1]$

$i = 2$

*Result* = *True*

**while**  $i \leq N$  and *Result* **do**

*Result* = *False*

    Build  $H_i$  and  $F_i$

**for** each configuration  $C(H_{i-1}, x)$  of  $F_{i-1}$  **do**

**for**  $j = 1$  to number of blocks of  $C(H_{i-1}, x)$  **do**

**if**  $i$  does not have any neighbor in the block  $j$  **then**

*Result* = *True*

$part = C(H_{i-1}, x)$

                insert  $i$  in the block  $j$  of  $part$

                generate the configuration  $C(H_i, y)$  corresponding to  $part$

**if**  $C(H_i, y)$  already exists **then**

$val(C(H_i, y)) = \min(val(C(H_i, y)), val(C(H_{i-1}, x)))$

**else**

$val(C(H_i, y)) = val(C(H_{i-1}, x))$

**end if**

**end if**

**end for**



```

if numberof blocks of  $C(H_{i-1}, x) < k$  then
     $Result = True$ 
     $part = C(H_{i-1}, x)$ 
    add to  $part$  a new block containing  $i$ 
     $val(part) = \max(val(C(H_{i-1}, x)), \text{number of blocks of } part)$ 
    generate the configuration  $C(H_i, y)$  corresponding to  $part$ 
    if  $C(H_i, y)$  already exists then
         $val(C(H_i, y)) = \min(val(C(H_i, y)), val(part))$ 
    else
         $val(C(H_i, y)) = val(part)$ 
    end if
end if
end for
 $i = i + 1$ 
end while

```

---



---

**Algorithm 3** Linear decomposition coloring algorithm (LDC)

---

**Input:** a graph  $G$

**Output:**  $k$  : the chromatic number of  $G$

```

Determine a lower bound (LB) of  $k$ 
Determine an upper bound (UB) of  $k$ 
while (LB  $\neq$  UB) do
    set the value of  $k$  between LB and UB (*)
     $result = k\text{-coloring}(G)$ 
    if  $result = false$  then
        LB =  $k + 1$ 
    else
        UB =  $k$ 
    end if
end while
(*)  $Kdic : k = \lfloor (LB + UB)/2 \rfloor, Kseq : k = LB + 1$ 

```

---

$C(H_i, 3) = [uw][vi]$  with value 2 or 3,  $C(H_i, 12) = [uw][v][i]$  with value 3,

$C(H_i, 14) = [uv][w][i]$  with value 3, and  $C(H_i, 10) = [u][vi][w]$  with value 3.

We cannot add a new block to  $C(H_{i-1}, 5)$ , because it has already got three blocks.

The gain may obviously be more significant when the size of the boundary set increases. Indeed, let  $T'(F_i)$  be the maximum number of partitions of  $F_i$  which have at most  $k$  blocks:

$$T'(F_i) = \sum_{j=1 \text{ to } k} A_{|F_i|, j}.$$

When no edge exists between the vertices of  $F_i$ , the gain  $Gain(F_i)$  for this step equals  $T(F_i)$  minus  $T'(F_i)$ :

$$Gain(F_i) = \sum_{j=k+1 \text{ to } |F_i|} A_{|F_i|,j}.$$

In the improved linear decomposition coloring algorithm (LDC), we start by determining a LB of the chromatic number of  $G$  (see Algorithm 3). For that, we apply on  $G$  some heuristics based on triangulated graphs. Indeed, it is easy to compute the chromatic number of a triangulated graph [27]. The used lower bound functions as follows: as long as the graph is not triangulated, we remove a vertex of smallest degree, and then we color the remaining triangulated graph by determining a perfect elimination order [27] on the vertices of  $G$ . The UB is obtained by applying the heuristic DSATUR [9]. Then we apply a succession of  $k$ -colorings,  $k$  varying between LB and UB. We consider two different versions of LDC, depending on the way the value of  $k$  is set: by dichotomy, denoted *Kdic*-LDC, or sequentially growing from LB to UB, denoted *Kseq*-LDC. In theory, *Kdic*-LDC has a better complexity than *Kseq*-LDC. Nevertheless, in practice *Kseq*-LDC gives good lower bounds of the chromatic number of hard instances that *Kdic*-LDC is unable to solve.

Algorithm 3 does not produce directly a coloring, nevertheless it can be obtained easily. All the configurations generated during the  $k$ -coloring have to be stored in memory. The color 1 is assigned to vertex  $N$ . When a  $k$ -coloring is found, a configuration of  $F_{N-1}$  corresponding to the configuration of  $F_N$  is chosen, a color is assigned to vertex  $N - 1$ , and so on until vertex 1 is colored. Assign colors to the vertices needs a lot of memory space but does not increase the time complexity of the algorithm.

### 3.6. The vertex linear ordering

The maximum size of the boundary sets  $F_{\max}(\mathcal{N})$  depends on the vertices numbering chosen (see Fig. 4). It corresponds to the linearwidth of the vertex linear ordering used to build the different subgraphs  $H_i$  (cf Section 2.3).

The complexity of the linear-decomposition is exponential with respect to  $F_{\max}(\mathcal{N})$ , so it is necessary to make a good choice when numbering the vertices of the graph. Unfortunately, finding an optimal vertex linear ordering in order to obtain the smallest linearwidth is a NP-complete problem [2]. A first method to reduce the size of the boundary sets is to number the vertices by applying a double BFS [27] on the graph. It produces the numbering  $\mathcal{N}_1$ . An other method is to begin the numbering from a clique and then order the vertices by decreasing number of edges with already numbered vertices. It produces the numbering  $\mathcal{N}_2$ . The aim of this method is to give the smallest numbers as possible to the vertices of boundary sets of size  $F_{\max}(\mathcal{N}_2)$ . To compare the linearwidths of these numberings with the linearwidth of the initial vertex numbering  $\mathcal{N}_0$ , we performed tests on random graphs  $G_{N,d}$  (graphs with  $N$  vertices and with density  $d$ ) and on COLOR02 computational symposium instances. Some representative results are reported in Table 2. For random graphs, results over average of 5 instances are given.

The numbering  $\mathcal{N}_2$  gave smaller  $F_{\max}(\mathcal{N})$  for more instances than the other numberings, so we use this ordering in experiments of Section 4. Despite this, the number of configurations increases sometimes very strongly when introducing a vertex with too few neighbors in the boundary set. To avoid this case we add a reduction on the graph before each  $k$ -coloring.

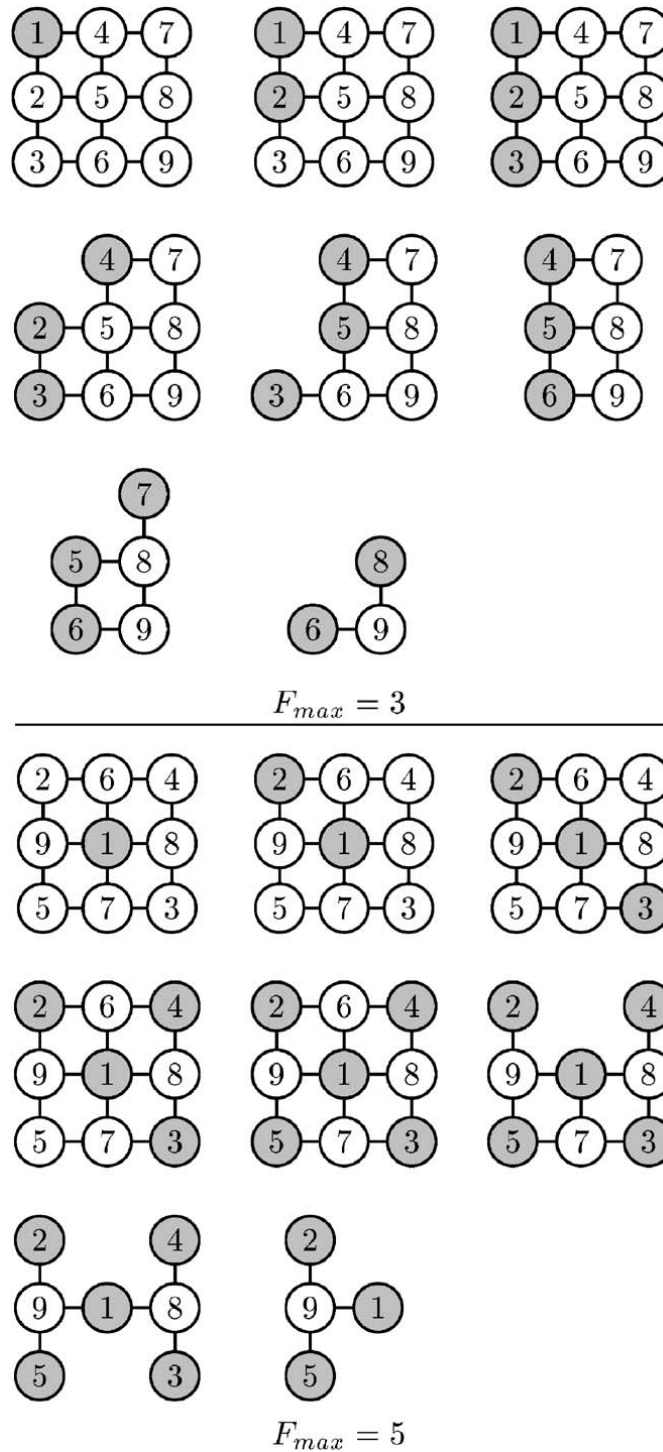


Fig. 4. The value of  $F_{max}$  is a consequence of the initial vertex numbering.

Table 2  
Comparison of different vertex linear orderings

Problem	$F_{\max}(\mathcal{N}_0)$	$F_{\max}(\mathcal{N}_1)$	$F_{\max}(\mathcal{N}_2)$	Problem	$F_{\max}(\mathcal{N}_0)$	$F_{\max}(\mathcal{N}_1)$	$F_{\max}(\mathcal{N}_2)$
G(30,0.1)	12.4	9.6	<b>8.4</b>	G(40,0.5)	34.0	34.4	<b>33.6</b>
G(40,0.1)	20.8	16.4	<b>15.0</b>	G(50,0.5)	43.4	43.4	<b>41.8</b>
G(50,0.1)	29.4	26.2	<b>22.0</b>	G(60,0.5)	53.8	53.8	<b>52.4</b>
G(60,0.1)	36.8	32.8	<b>28.0</b>	G(30,0.9)	27.8	27.8	<b>27.2</b>
G(70,0.1)	44.0	43.0	<b>34.2</b>	G(40,0.9)	37.8	37.8	<b>36.4</b>
G(80,0.1)	55.0	50.8	<b>42.8</b>	G(50,0.9)	47.4	47.4	<b>46.8</b>
G(90,0.1)	63.6	60.0	<b>51.8</b>	G(60,0.9)	57.8	57.8	<b>57.0</b>
G(30,0.5)	24.8	24.6	<b>22.6</b>	G(70,0.9)	67.6	67.4	<b>67.0</b>
Mug88_1	20	13	<b>8</b>	Mug88_25	20	13	<b>8</b>
Mug100_1	17	22	<b>7</b>	Mug100_25	19	16	<b>8</b>
1-FullIns4	61	71	<b>46</b>	1-FullIns5	187	216	<b>132</b>
2-FullIns3	37	28	<b>22</b>	2-FullIns4	157	105	<b>93</b>
2-FullIns5	637	417	<b>359</b>	3-FullIns3	61	45	<b>36</b>
3-FullIns4	321	247	<b>200</b>	4-FullIns3	91	57	<b>49</b>
4-FullIns4	571	332	<b>302</b>	5-FullIns3	127	127	<b>64</b>
5-FullIns4	925	772	<b>447</b>	1-Insert_4	<b>31</b>	40	32
1-Insert_5	98	145	<b>91</b>	1-Insert_6	300	406	<b>276</b>
2-Insert_4	<b>48</b>	56	52	2-Insert_5	<b>197</b>	251	240
3-Insert_3	<b>13</b>	<b>13</b>	16	3-Insert_4	<b>69</b>	92	84
3-Insert_5	<b>350</b>	578	436	4-Insert_3	<b>15</b>	<b>15</b>	20
4-Insert_4	<b>94</b>	120	124	Miles250	67	48	<b>16</b>
Le450_5a	381	368	<b>339</b>	Le450_5b	386	382	<b>343</b>
Le450_5c	397	401	<b>385</b>	Le450_5d	400	403	<b>385</b>
Le450_15a	373	365	<b>339</b>	Le450_15b	378	372	<b>336</b>
Le450_15c	420	418	<b>396</b>	Le450_15d	416	415	<b>401</b>
Le450_25a	351	348	<b>293</b>	Le450_25c	416	412	<b>394</b>
Le450_25d	411	409	<b>384</b>				

### 3.7. Vertex reduction

Before each  $k$ -coloring, we delete some vertices of the graph by using the following property: for each vertex  $x$ , if the degree of  $x$  is strictly lower than  $k$ ,  $x$  and its adjacent edges can be erased from the graph [12]. Indeed, assume  $x$  has  $k - 1$  neighbors. In the worst case, those neighbors must have different colors. Then the vertex  $x$  can take the  $k$ th color. It does not interfere in the coloring of the remaining vertices because all its neighbors have already been colored. Therefore we can consider from the beginning that it will take a color unused by its neighbors and delete it from the graph before the coloring. We apply this principle recursively by examining the remaining vertices until having totally reduced the graph or being unable to delete any other vertex.

## 4. Experimental results

Our LDC algorithms have been implemented on a PC AMD Athlon Xp 2000+ in C language. We performed tests on random graphs and on benchmark instances used at the computational symposium COLOR02.

Table 3  
Results on random graphs

Problem	$M$	LB	UB	LDC			
				$Kdic$	$T$	$Kseq$	$T$
G(30,0.1)	43.8	2.8	3.0	3.0	0.00	3.0	0.00
G(40,0.1)	75.8	3.0	3.6	3.2	0.00	3.2	0.00
G(50,0.1)	123.2	3.0	4.2	4.0	0.00	4.0	0.78
G(60,0.1)	177.0	3.0	4.6	4.0[2]	0.00	4.0	2.54
G(70,0.1)	237.8	3.0	5.0			4.0	2.66
G(80,0.1)	320.4	3.0	5.0			4.2[4]	103.16
G(90,0.1)	397.0	3.2	5.6			4.6[3]	104.80
G(100,0.1)	477.2	3.0	5.6			5.0[4]	190.99
G(110,0.1)	604.4	3.8	6.0			5.0[4]	91.90
G(120,0.1)	711.8	3.2	6.4			[0]	*
G(30,0.5)	211.2	5.2	8.2	7.0	11.18	7.0	2.92
G(40,0.5)	380.0	6.2	9.8	8.2	6.19	8.2	6.16
G(50,0.5)	608.6	6.2	12.0	6.0[3]	97.93	9.4[3]	83.12
G(60,0.5)	881.8	6.6	12.8	[0]	*	[0]	*
G(30,0.9)	392	15.2	16.2	15.8	0.02	15.8	0.01
G(40,0.9)	699.8	18.2	22.6	18.8	7.11	18.8	0.44
G(50,0.9)	1103.6	20.8	25.4	22.8	13.2	22.8	10.79
G(60,0.9)	1606.6	24.0	30.6	[0]	*	26.2[4]	105.02
G(70,0.9)	2172.6	25.0	33.4			28[1]	684.8

#### 4.1. Random graphs

Random graphs  $G_{N,d}$  have been created. For each graph,  $N$  is the number of vertices and  $d$  its density. We generated graphs with densities 0.1, 0.5 and 0.9. We give the results of our experiments over average of 5 instances in Table 3. For each type of graph, we give the number of edges  $M$  and the values LB and UB of our initial bounds. Algorithms  $Kdic$ -LDC and  $Kseq$ -LDC have been applied to each instance, using a CPU time limit of half an hour. For each algorithm, we give the average chromatic number and the CPU time when the program is able to compute. Otherwise, when less than 5 instances have been solved, we indicate the number of solved instances in brackets. Starting from  $N = 30$ , we increase the size of random graphs by step 10 until less than 3 instances are solved.

Our algorithms are not very efficient on random graphs in comparison with the results of other exact methods [20,23,24]. Indeed, Desrosiers, Galinier and Hertz are able to solve graphs with density 0.5 and with up to 100 vertices. This is mainly due to the fact that edges repartition in the graph is homogeneous, inducing a large linearwidth. For a graph with 50 vertices, we have (see Table 2)  $F_{\max}(\mathcal{N}_2) = 41$  when the graph density is 0.5 and  $F_{\max}(\mathcal{N}_2) = 46$  when the graph density is 0.9. Nevertheless, results of Table 3 reflect clearly the possibilities and the limits of our method. We are able to solve graphs with density 0.9 and 60 vertices. These graphs have a very large linearwidth but are also very constrained: at each step only few configurations are generated despite the big size of the boundary sets. The decomposition

Table 4  
Lower bound and Upper bound results

Problem	$N$	$M$	LB	UB	Problem	$N$	$M$	LB	UB
Fpsol2.i.1	496	11654	<b>65</b>	<b>65</b>	1-FullIns3	30	100	3	5
Fpsol2.i.2	451	8691	<b>30</b>	<b>30</b>	1-FullIns4	93	593	3	6
Fpsol2.i.3	425	8688	<b>30</b>	<b>30</b>	1-FullIns5	282	3247	3	8
Inithx.i.1	864	18707	<b>54</b>	<b>54</b>	2-FullIns3	52	201	4	5
Inithx.i.2	645	13979	<b>31</b>	<b>31</b>	2-FullIns4	212	1621	4	6
Inithx.i.3	621	13969	<b>31</b>	<b>31</b>	2-FullIns5	852	12201	4	7
Le450-5a	450	5714	5	11	3-FullIns3	80	346	5	6
Le450-5b	450	5734	5	11	3-FullIns4	405	3524	2	7
Le450-5c	450	9803	4	13	4-FullIns3	114	541	6	7
Le450-5d	450	9757	4	12	4-FullIns4	690	6650	2	8
Le450-15a	450	8168	13	18	5-FullIns3	154	792	7	8
Le450-15b	450	8169	15	17	5-FullIns4	1085	11395	2	9
Le450-15c	450	16680	11	25	1-Inser_4	67	232	2	5
Le450-15d	450	16750	11	26	1-Inser_5	202	1227	2	6
Le450-25a	450	8260	20	25	1-Inser_6	607	6337	2	7
Le450_25b	450	8263	<b>25</b>	<b>25</b>	2-Inser_3	37	72	2	4
Le450-25c	450	17343	22	30	2-Inser_4	149	541	2	5
Le450-25d	450	17425	19	29	2-Inser_5	597	3936	2	6
Mulsol.i.1	197	3925	<b>49</b>	<b>49</b>	3-Inser_3	56	110	2	4
Mulsol.i.2	188	3885	<b>31</b>	<b>31</b>	3-Inser_4	281	1046	2	5
Mulsol.i.3	184	3916	<b>31</b>	<b>31</b>	3-Inser_5	1406	9695	2	6
Mulsol.i.4	185	3946	<b>31</b>	<b>31</b>	4-Inser_3	79	156	2	4
Mulsol.i.5	186	3973	<b>31</b>	<b>31</b>	4-Inser_4	475	1795	2	5
School1	385	19095	<b>14</b>	<b>14</b>	Mug100-1	100	166	3	4
School1_nsh	352	14612	<b>14</b>	<b>14</b>	Mug100-25	100	166	3	4
Zeroin.i.1	211	4100	<b>49</b>	<b>49</b>	Mug88-1	88	146	3	4
Zeroin.i.2	211	3541	<b>30</b>	<b>30</b>	Mug88-25	88	146	3	4
Zeroin.i.3	206	3540	<b>30</b>	<b>30</b>	Games120	120	638	<b>9</b>	<b>9</b>
Anna	138	493	<b>11</b>	<b>11</b>	Miles250	128	387	7	8
David	87	812	<b>11</b>	<b>11</b>	Miles500	128	2340	<b>20</b>	<b>20</b>
Homer	561	1629	<b>13</b>	<b>13</b>	Miles750	128	4226	<b>31</b>	<b>31</b>
Huck	74	602	<b>11</b>	<b>11</b>	Miles1000	128	6432	<b>42</b>	<b>42</b>
Jean	80	508	<b>10</b>	<b>10</b>	Miles1500	128	10396	<b>73</b>	<b>73</b>

method is not efficient on random graphs with density 0.5. Indeed, the large size of the linearwidth induces the possibility to generate an exponential number of configurations and the 0.5 density does not limit enough the number of valid configurations. Our linear-decomposition method works better on graphs with small density. Indeed, the maximum size of the boundary set can be very reduced by the vertex numbering  $\mathcal{N}_2$ . For this kind of random graphs, we are able to solve instances with up to 100 vertices.

#### 4.2. DIMACS and COLOR02 instances

We performed tests on benchmark instances used at the computational symposium COLOR02, among which well-known DIMACS instances (see description of the instances at <http://mat.gsia.cmu.edu/COLOR02>).

Table 5  
Results on COLOR02 instances

Problem	LDC			B&C		VCS	
	$F_{\max}$	$k$	$T$	$k$	$T$	$k$	$btk$
Mug88_1	8	4	0.0	4	485	4	2204467
Mug88_25	8	4	0.0	4	1690	4	942961
Mug100_1	7	4	0.1	4	4029	4	1406570
Mug100_25	8	4	0.0	4	5498	4	974170
1-FullIns4	27	5	7.3	5	703	5	6
1-FullIns5	26	4–6	*	4–6	*	6	271
2-FullIns3	9	5	0.0	5	3	5	1
2-FullIns4	19	5–6	*	5–6	*	6	8
2-FullIns5	20	5–7	*	5–7	*	7	715
3-FullIns3	11	6	0.0	6	1	6	0
3-FullIns4	20	3–7	*	6–7	*	7	10
4-FullIns3	11	7	0.0	7	3	7	1
4-FullIns4	17	4–8	*	7–8	*	8	12
5-FullIns3	13	8	0.0	8	3	8	1
5-FullIns4	17	3–9	*	7–9	*		
1-Inser_4	21	4–5	*	2–5	*	5	104296036
1-Inser_5	20	3–6	*	4–6	*	*	(133727661)
1-Inser_6	18	3–7	*	4–7	*	*	(50929137)
2-Inser_4	28	3–5	*	4–5	*	*	(154902785)
2-Inser_5	18	3–6	*	3–6	*	*	(48458541)
3-Inser_3	16	4	23.3	4	10	4	723616
3-Inser_4	28	3–5	*	3–5	*	*	(95076991)
3-Inser_5	17	3–6	*	3–6	*	*	(13784327)
<b>4-Inser_3</b>	<b>20</b>	<b>4</b>	<b>1774</b>	<b>3–4</b>	<b>*</b>	<b>*</b>	<b>(228367528)</b>
4-Inser_4	27	3–5	*	3–5	*	*	(70891706)
Miles250	9	8	0.0			8	0
Le450_5a	15	5–9	*	5–9	*	5	0
Le450_5b	16	5–9	*	5–9	*	5	0
Le450_5c	22	5–6	*			5	0
Le450_5d	46	5–7	*	5–10	*	5	0
Le450_15a	21	15–18	*	15–17	*	15	0
Le450_15b	21	15–17	*	15–17	*	15	0
Le450_15c	21	15–25	*	15–24	*	15	0
Le450_15d	21	15–26	*	15–23	*	15	0
Le450_25a	25	25	0.0			25	0
Le450_25c	30	25–28	*	25–28	*	25	0
Le450_25d	30	25–28	*	25–28	*	25	0

We first apply on each instance the lower bound and upper bound described in Section 3.5. Results are reported in Table 4. For each graph we give the number of vertices  $N$ , the number of edges  $M$ , and the values LB and UB of our initial bounds. At this stage, the LB equals the UB for some of the DIMACS instances, so we did not apply any  $k$ -coloring algorithm on these graphs.

We compare our LDC algorithms with the recent results of the exact Branch-and-Cut algorithm of Mendez Diaz and Zabala [22]. We use a CPU time limit of half an hour. The results of our experiments



are reported in Table 5. The time column for B&C is only indicative since the results are from different machines. They used C++ code using ABACUS framework and CPLEX 6.0 LP solver on a Sun ULTRA workstation. Their CPU time limit was two hours. Comparing their time results for the same implementation of *Dsat* that we use, our computer seems to run three times faster than their computer. We report also the results of Desrosiers et al. [24] in *VCS* columns. Column *btk* contains the number of backtracks needed by their exact algorithm to solve an instance reduced to a vertex-critical subgraph. Values are enclosed in parentheses when their algorithm is not able to find the chromatic number within the time limit.

The  $F_{\max}$  column indicates for each graph the maximal value of the boundary set reached by our algorithm. We give the chromatic number of the graph and the CPU time when the program is able to compute. Otherwise, asterisks in the *time* column indicate that the program exceeded the time limit. In this case, we report the best lower and upper bounds of the chromatic number obtained by the algorithm.

The results of *Kseq*-LDC are equivalent or better than those of *Kdic*-LDC on the tested instances, that is why we report only *Kseq*-LDC results. Our algorithm runs very fast on *mug* instances (0.0 or 0.1 s), because the vertex linear ordering chosen provides small linearwidths for these graphs ( $F_{\max}$  column). For these instances, the results of *Kdic*-LDC and *Kseq*-LDC are the same, since  $UB = LB + 1$ . B&C and *VCS* found the chromatic number, but spent time to explore the branch-and-cut tree. Our algorithms are also able to solve some instances which have a large linearwidth, because all the partitions of the boundary set have not necessarily to be generated. We found the chromatic number of 13 instances, 12 of which solved in less than 30 s. The LDC algorithm is the first to our knowledge to find the chromatic number of the instance *4-Inser\_3* (bolded in Table 5) with an exact method.

## 5. Conclusions

In this paper, we have presented an original method to solve the graph coloring problem by an exact way. This method has the advantage of solving easily large instances which have a small linearwidth, such as *mug* instances, and allows us to solve the difficult instance *4-Inser\_3*. We consider using the linear-decomposition mixed with heuristics approach to deal with unbounded linearwidth instances.

## References

- [1] Robertson N, Seymour P. Graph minors. ii. Algorithmic aspects of tree-width. *Journal of Algorithms* 1986;7:309–22.
- [2] Bodlaender HL. A tourist guide through treewidth. *Acta Cybernetica* 1993;11:1–21.
- [3] Carlier J, Lucet C. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics* 1996;65:141–56.
- [4] Lucet C. Méthode de décomposition pour l'évaluation de la fiabilité des réseaux. PhD thesis, Université de Technologie de Compiègne, 1993.
- [5] Manouvrier J. Méthode de décomposition pour résoudre des problèmes combinatoires sur les graphes. PhD thesis, Université de Technologie de Compiègne, 1998.
- [6] de Werra D. An introduction to timetabling. *European Journal of Operation Research* 1985;19:151–62.
- [7] de Werra D. On a multiconstrained model for chromatic scheduling. *Discrete Applied Mathematics* 1999;94:171–80.
- [8] Garey MR, Johnson DS. *Computers and intractability—A guide to the theory of NP-completeness*. San Francisco: Freeman; 1979.
- [9] Brelaz D. New methods to color the vertices of a graph. *Communications of the ACM* 1979;22(4):251–6.
- [10] Funabiki N, Higashino T. A minimal-state processing search algorithm for graph coloring problems. *IEICE Transactions on Fundamentals* 2000;E83-A(7):1420–30.



- [11] Galinier P, Hao JK. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization* 1999;3(4):379–97.
- [12] Glover F, Parker M, Ryan J. Coloring by tabu branch and bound. In: Trick and Johnson, Cliques, Coloring, and Satisfiability: Proceedings of the Second DIMACS Implementation Challenge. American Mathematical Society, 1993, p. 285–308.
- [13] Hertz A, Werra DD. Using tabu search techniques for graph coloring. *Computing* 1987;39:345–51.
- [14] Manvel B. Extremely greedy coloring algorithms. In: Harary F, Maybee JS, editors. *Graphs and Applications: Proceedings of the First Colorado Symposium on Graph Theory*. New York: Wiley; 1985. p. 257–70.
- [15] Morgenstern CA. Distributed coloration neighborhood search. In: Trick and Johnson, Cliques, Coloring, and Satisfiability: Proceedings of the Second DIMACS Implementation Challenge. American Mathematical Society, 1993, p. 335–57.
- [16] Sarma SS, Bandyopadhyay SK. Some sequential graph colouring algorithms. *International Journal of Electronic* 1989;67(2):187–99.
- [17] Kubale M, Jackowski B. A generalized implicit enumeration algorithm for graph coloring. *Communications of the ACM* 1985;28(4):412–8.
- [18] Sager TJ, Lin S. A pruning procedure for exact graph coloring. *ORSA Journal on Computing* 1991;3:226–30.
- [19] Sewell E. An improved algorithm for exact graph coloring. In: Trick and Johnson Cliques, Coloring, and Satisfiability: Proceedings of the Second DIMACS Implementation Challenge. American Mathematical Society, 1993, p. 359–373.
- [20] Mehrotra A, Trick MA. A column generation approach for graph coloring. *INFORMS Journal on Computing* 1996;8(4):344–54.
- [21] Diaz IM, Zabala P. A branch-and-cut algorithm for graph coloring. In: *Computational Symposium on Graph Coloring and its Generalizations (COLOR02)*. Ithaca, New York, September 2002.
- [22] Diaz IM, Zabala P. A branch-and-cut algorithm for graph coloring. *Optimization Online: Combinatorial Optimization*, September 2002.
- [23] Herrmann F, Hertz A. Finding the chromatic number by means of critical graphs. *ACM Journal of Experimental Algorithmics* 2002;7(10):1–9.
- [24] Desrosiers C, Galinier P, Hertz A. Efficient algorithms for finding critical subgraphs. *Cahiers du GERAD G-2004-31*, 2004.
- [25] Amborg DGCS, Proskurowski A. Complexity of finding embeddings in a  $k$ -tree. *SIAM Journal on Algebraic and Discrete Methods* 1987;8:277–84.
- [26] Bodlaender HL. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing* 1996;25(6):1305–17.
- [27] Golumbic MC. *Algorithmic graph theory and perfect graphs*. New York: Academic Press; 1980.

# Upper and Lower Bounds for the Minimum Sum Coloring Problem

Moukrim A.<sup>a,1</sup>, Sghiouer K.<sup>a,2</sup>, Lucet C.<sup>b,3</sup> and Li Y.<sup>b,4</sup>

<sup>a</sup>*Université de Technologie de Compiègne, HeuDiaSyC UMR UTC-CNRS 6599,  
BP 20529, 60205 Compiègne, France*

<sup>b</sup>*Université de Picardie Jules Verne, M.I.S. EA 4290, 33 rue Saint Leu, 80000  
Amiens, France*

---

## Abstract

In this paper we look at the Minimum Sum Coloring Problem (MSCP). MSCP is a graph-coloring problem whose goal is to minimize the sum of colors, where colors are represented by natural numbers. MSCP has applications in VLSI design, scheduling and resource allocation. We propose calculating upper and lower bounds for MSCP using a single coloring algorithm: a Memetic Algorithm (MA) hybridizing a simple genetic algorithm with local search. An upper bound can be obtained by coloring the graph with this algorithm, while a general lower bound can be produced by coloring optimally some partial graphs extracted from the original graph, in particular a partial graph generated by partitioning the original graph into cliques. In this context we define a new problem for obtaining this kind of general lower bound, namely Partition into Cliques for MSCP (PCMSCP), and prove its NP-completeness. Since PCMSCP can easily be transformed into a coloring problem by virtue of certain properties of graphs, we are able to use the same algorithm to obtain upper bounds for PCMSCP and lower bounds for MSCP. We can consequently compute both upper and lower bounds by obtaining valid solutions to MSCP and PCMSCP respectively. Experimental results show that our approach strictly improves most upper and lower bounds for the instances in [8]. We also present the results for some instances from the DIMACS library [6] in order to provide a basis for future

work. The upper and lower bounds obtained allow us to identify the optimum for 26 instances out of the 76 tested.

*Keywords:* Minimum Sum Coloring, Lower bounds, Local search, Memetic Algorithm.

---

## 1 Introduction

The Minimum Sum Coloring Problem (MSCP) was first introduced by Kubicka and Schwenk, who also proved its NP-completeness [10]. MSCP is a graph-coloring problem whose goal is to minimize the sum of colors, where the colors are represented by natural numbers. It is closely related to the basic Graph Coloring Problem (GCP), whose goal is to minimize the number of colors. MSCP has applications in VLSI design, scheduling and resource allocation [15].

GCP has been extensively studied in the literature, and many good results have been produced [3,5,14,16]. These studies can be divided into two categories. The first category consists of exact methods such as branch and bound and decomposition [14], that give an optimal solution for very small graphs or specific families of graphs. The second category involves basic heuristics such as greedy algorithms [2,12] and meta-heuristics [5,3]. Recently we proposed an adjusted greedy algorithm MRLF [13] in  $O(n^3)$ , which improves the two well known existing greedy algorithms DSATUR and RLF for MSCP.

The main purpose of this paper is to propose an approach for calculating both upper and lower bounds for MSCP. The MSCP literature features some theoretical results regarding upper and lower bounds for specific graph classes [1,9,10,19], but to our knowledge the only numerical results for the general case have been obtained by a parallel genetic algorithm proposed in [8].

In our approach upper and lower bounds for MSCP can be calculated by a same-coloring algorithm. This algorithm, known as the Memetic Algorithm for MSCP (MA-MSCP), is a simple hybrid genetic algorithm based on a crossover operator proposed in [3] and using local search techniques for mutation.

An upper bound can be obtained by coloring the graph with this algorithm. Obtaining a general lower bound involves extracting partial graphs for which

---

<sup>1</sup> Email: aziz.moukrim@utc.fr

<sup>2</sup> Email: kaoutar.sghiouer@utc.fr

<sup>3</sup> Email: corinne.lucet@u-picardie.fr

<sup>4</sup> Email: yu.li@u-picardie.fr

<sup>5</sup> This research was partially supported by *le Conseil Régional de Picardie*.

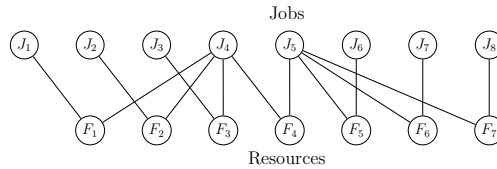


Fig. 1. Scheduling Problem

the optimal solution can easily be computed, and in particular partial graphs generated by partitioning the graph into cliques. This has led us to introduce a new problem in relation to the lower bound, namely Partition into Cliques for MSCP (PCMSCP) and to prove its NP-completeness. Since PCMSCP can easily be transformed into the problem of coloring the complementary graph of the original graph, we use the same coloring algorithm MA-MSCP to solve PCMSCP by considering only its objective function. We therefore compute both the upper and lower bounds, by obtaining valid solutions for MSCP and PCMSCP respectively.

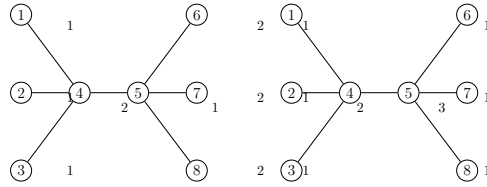
MA-MSCP strictly improves most upper and lower bounds for the instances in [8], and also gives results for some instances in the DIMACS library [6], thus providing a basis for future work. Moreover, the upper and lower bounds obtained mean that the optimum of the 26 instances among the 76 tested can be identified.

The paper is organized as follows. In the next section we give a formal definition for MSCP as well as GCP. In Section 3 we study some general lower bounds for MSCP and then introduce PCMSCP. In Section 4 we present the MA-MSCP approach for obtaining upper and lower bounds for MSCP. In Section 5 we present and analyze experimental results.

## 2 Minimum Sum Coloring Problem

MSCP has recently begun to attract attention, because in a modern distributed system the average completion time of a job is an important measure of quality of service.

Consider a distributed system where a set of jobs will be executed by a cluster of processors. Let us denote the jobs to be executed as  $J_1, J_2, \dots, J_n$ , and the resources used by these jobs as  $F_1, F_2, \dots, F_t$  (Fig. 1). For simplicity, we suppose that each job takes one unit of time and exclusively accesses certain resources in accomplishing its work. We also suppose that there is no limit on the number of processors. These different constraints are modeled by a graph with vertices representing jobs, and edges representing access conflicts to resources. If we color the graph so that two adjacent vertices have different



(a) Optimal solution for GCP. (b) Optimal solution for MSCP.

$$\Sigma(G, c) = 12, \chi(G) = 2 \qquad \Sigma(G) = 11, s(G) = 3$$

Fig. 2. Comparison of GCP and MSCP on trees.

colors, a color can represent a period where the corresponding jobs can run together. The number of colors thus corresponds to the time taken by the cluster to execute all jobs. Furthermore, if we represent colors by the natural numbers  $1, 2, \dots, k$ , the sum of colors corresponds to the total completion time of all jobs, and consequently to the average time for executing a job.

When coloring a graph, if the goal is to minimize the number of colors then we are dealing with the basic Graph Coloring Problem (GCP), whereas if the objective is to minimize the sum of colors then we are dealing with the Minimum Sum Coloring Problem (MSCP).

We consider an undirected graph  $G = (V, E)$ , where  $V$  is the set of  $|V| = n$  vertices and  $E$  the set of  $|E| = m$  edges.

A *coloring* of  $G$  is a function  $c : v \mapsto c(v)$  that assigns to each vertex  $v$  a color  $c(v)$ . A coloring is said to be *feasible* if, for any pair of vertices  $u, v \in V$  such that  $[u, v] \in E$ , we have  $c(u) \neq c(v)$ . If  $k$  colors are used in a feasible coloring, then this coloring of  $G$  is called a  $k$ -*coloring*. The minimum value of  $k$  among all the feasible colorings is called the *chromatic number* of the graph, denoted as  $\chi(G)$ . GCP involves finding this minimum number of colors.

Equivalently, a coloring can be seen as a *partition*  $X$  of the set of vertices into  $k$  independent subsets, called *color classes*:  $X_1, \dots, X_k$ , where the vertices in  $X_i$  are colored with color  $i$ . The number of vertices in  $X_i$  is denoted  $x_i$ . If a feasible coloring of  $G$  is denoted  $X$ , with the respective color classes  $X_1, \dots, X_k$  ordered in decreasing order of size, the associated sum of the colors can be written as follows:

$$\Sigma(G, c) = 1.x_1 + 2.x_2 + \dots + k.x_k$$

MSCP consists in finding a feasible coloring of the graph as previously described, such that the sum of the colors has the lowest possible value.

This optimal value is called the *chromatic sum* of  $G$  and denoted  $\sum(G)$ . The smallest number of colors required by an optimal solution of MSCP is called the *chromatic strength* of the graph, and is denoted  $s(G)$ . The *chromatic strength* of  $G$  can be greater than its *chromatic number*.

As an example, the graph  $G$  in Figure 2(a) is a tree, so  $\chi(G) = 2$ . The best solution for MSCP using two colors has a sum coloring which is equal to 12. However,  $s(G) = 3$  and  $\sum(G) = 11$  (see Figure 2(b)).

There are some theoretical results in relation to upper and lower bounds for MSCP, as well as properties for restricted graph families such as chain bipartite graphs, interval graphs and  $k$ -split-graphs [19]. However, to our knowledge, little work has been done regarding general bounds for the chromatic sum. In [20] the authors show that the chromatic sum of any graph  $G$  with  $n$  vertices,  $m$  edges and a chromatic number  $\chi(G)$  is bounded as follows :

$$\sum(G) \leq n + m$$

$$\lceil \sqrt{8m} \rceil \leq \sum(G) \leq \left\lfloor \frac{3(m+1)}{2} \right\rfloor$$

$$n + \frac{\chi(G)(\chi(G) - 1)}{2} \leq \sum(G) \leq \frac{n(\chi(G) + 1)}{2}$$

These theoretical bounds can be attained for specific graph classes, but are still a long way from the optimal solution in the general case. To overcome this we propose an algorithmic solution for the upper and lower bounds of MSCP.

### 3 Lower Bounds for MSCP

A *partial graph*  $G' = (V, E')$  of  $G = (V, E)$  is a graph where  $E'$  is a subset of  $E$ . Any feasible coloring of  $G$  is a feasible coloring of  $G'$ . Therefore, the following result immediately holds.

**Property 1** *If  $G'$  is a partial graph of  $G$ , then the chromatic sum of  $G'$  is a lower bound for the chromatic sum of  $G$ .*

To calculate a lower bound of MSCP we need to find some partial graphs whose chromatic sums can be efficiently computed. In a subsequent section we present two families of partial graphs for which an optimal solution can be determined polynomially.

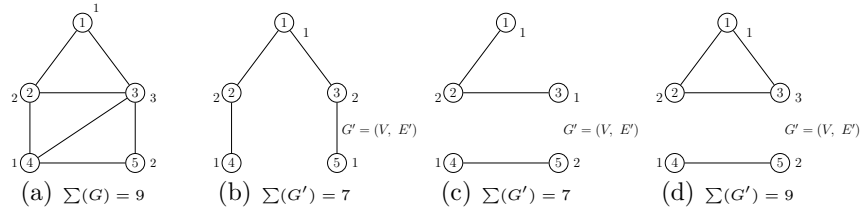


Fig. 3. A graph  $G$  (a), some partial graphs of  $G$  and their associated chromatic sum (b)(c)(d).

### 3.1 Bipartite partial graphs

In this section we are interested in finding a bipartite partial graph  $B$  of any graph  $G = (V, E)$ , such that  $\Sigma(B)$  is maximum over all the possible bipartite partial graphs of  $G$ . The optimal solution of this problem will be denoted  $\Sigma_{bipart}(G)$ .

Since MSCP is already NP-complete for bipartite graphs [7], computing  $\Sigma_{bipart}(G)$  is also an NP-complete problem. Despite this, there are two sub-families of bipartite graphs for which the chromatic sum can be optimally computed, namely paths and trees.

The optimal solution for the MSCP of a tree is obtained in linear time [9]. Any spanning tree  $T = (V, E')$  of a graph  $G = (V, E)$  ( $E' \subseteq E$ ) is a partial graph, and so its chromatic sum  $\Sigma(T)$  is a lower bound of the chromatic sum of  $G$ . The quality of this lower bound depends on the spanning tree used. As an example, consider the graph  $G = (V, E)$  in Figure 3(a). The chromatic sum of this graph's spanning tree  $T_1 = (V, \{[1, 3], [2, 3], [3, 4], [3, 5]\})$  is 6, whereas the chromatic sum of the spanning tree in Figure 3(b) is 7. The aim is to find a spanning tree  $T$  such that  $\Sigma(T)$  is maximum over all the possible spanning trees of  $G$ . The optimal solution of this problem will be denoted  $\Sigma_{tree}(G)$ .

Now, if we consider a path  $P$  of length  $l$ , with  $n$  vertices, then  $l = n - 1$ . By using colors 1 and 2, we get a feasible coloring of  $P$  with  $\lceil \frac{n}{2} \rceil$  vertices colored by 1. The chromatic sum  $\Sigma(P)$  is therefore  $\lceil \frac{n}{2} \rceil + 2 \lfloor \frac{n}{2} \rfloor$ . Moreover, any graph  $G = (V, E)$  can be partitioned into disjoint paths  $p_1, \dots, p_k$  such that  $\bigcup V(p_i) = V$ , where  $V(p_i)$  is the vertex set of path  $p_i$ . The graph  $p_1 \cup \dots \cup p_k$  is a partial graph of  $G$  whose chromatic sum is  $\Sigma(p_1) + \dots + \Sigma(p_k)$ . Accordingly, considering partial graphs that are unions of paths is a possible means of obtaining a lower bound for MSCP (see Figure 3(c)).

For an arbitrary graph  $G = (V, E)$ , Partition into Paths for MSCP consists in finding a partial graph  $P$  that is a union of paths in  $G$ , such that  $\Sigma(P)$

is maximum over all the possible partitions into paths of  $G$ . The optimal solution of this problem is denoted  $\sum_{dpaths}(G)$ .

**Proposition 3.1** *Let  $G = (V, E)$  be an arbitrary graph.  $\sum_{dpaths}(G)$  can be computed in polynomial time.*

**Proof.** A partition into paths can be seen as a matching of the graph by simply removing one edge out of two in every path. The partial graph is thus composed of  $\alpha$  edges whose vertices are colored with 1 and 2 and  $\beta$  isolated vertices colored with 1. Therefore, the chromatic sum is  $3\alpha + \beta$ . Partitioning the graph to obtain the best lower bound of MSCP is equivalent to seeking a partition with a maximum number of edges, in other words to solving the maximum matching problem. This problem is solvable in polynomial time [11].  $\square$

**Proposition 3.2** *For any graph  $G$ , we have  $\sum_{dpaths}(G) \leq \sum_{tree}(G)$ .*

**Proof.** Let  $G = (V, E)$  be a graph and  $P_{opt}$  be an optimal partition into paths of  $G$ . We introduce a forest  $F$  that we initialize by  $P_{opt}$ . Since  $G$  is connected, there are at least two connected components  $T_1$  and  $T_2$  in  $F$  such that there are two vertices  $x$  in  $T_1$  and  $y$  in  $T_2$  with  $[x, y]$  belonging to  $E$ . Adding this edge to  $F$  does not create any cycles, and  $F$  remains a forest. Similarly,  $F$  is transformed into a spanning tree  $T$  of  $G$  without any edges being removed from  $P_{opt}$ . Therefore,  $\sum_{dpaths}(G) = \Sigma(P_{opt}) \leq \Sigma(T) \leq \sum_{tree}(G)$ , and the result holds.  $\square$

**Proposition 3.3** *For an arbitrary graph  $G$ ,*

$$n \leq \sum_{dpaths}(G) \leq \sum_{tree}(G) \leq \sum_{bipart}(G) \leq MAX_{LBbipart}$$

where  $MAX_{LBbipart} = n + \lfloor n/2 \rfloor$ .

**Proof.** In view of the previous results, only the final inequality requires a proof. Because a bipartite graph has a vertex set  $V$  divided into two independent sets  $V_1$  and  $V_2$ , all vertices in  $V_1$  can have color 1 and all vertices in  $V_2$  can have color 2. In the worst case  $|V_2| = \lfloor n/2 \rfloor$ . Therefore, using 2 colors, the chromatic sum cannot exceed  $\lfloor n/2 \rfloor + 2\lfloor n/2 \rfloor$ , hence  $\sum_{bipart}(G) \leq \lfloor n/2 \rfloor + 2\lfloor n/2 \rfloor = n + \lfloor n/2 \rfloor$ .  $\square$

In the next section we propose another family of partial graphs, where more constraints are preserved, to evaluate lower bounds for MSCP, and we then compare the lower bounds thus obtained with  $MAX_{LBbipart}$ .



### 3.2 Partition into cliques

If we consider a clique of size  $k$ , there exists only one way of coloring that requires  $k$  colors. The associated sum of colors is  $k(k+1)/2$ . It is therefore possible to evaluate an optimal clique for MSCP.

Suppose that the vertex set  $V$  is partitioned into  $V_1, V_2, \dots, V_l$  such that the subgraph induced by  $V_i$ ,  $G(V_i)$  is a clique. The graph  $G(V_1) \cup G(V_2) \cup \dots \cup G(V_l)$  is a partial graph of  $G$ , and its chromatic sum is  $\sum_1^l \frac{|V_i|(|V_i|+1)}{2}$ . This is a lower bound of MSCP for  $G$ .

For a good approximation of MSCP we therefore have to find a partition into cliques, such that the associated chromatic sum is as large as possible. This problem is an optimization problem that we term Partition into Cliques for MSCP (PCMSCP). Clearly, this problem is closely related to the well known Partition into Cliques Problem (PCP), but the two problems do not necessarily have the same optimal solution. In fact, if the minimum number of cliques is  $k$  for PCP, the optimal solution for PCMSCP may contain more than  $k$  cliques. In the example in Figure 4(b), the partition  $\Lambda_1 = \{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\}\}$  is the optimal solution for PCP.  $\Lambda_1$  is composed of three cliques and its chromatic sum is  $3 \cdot \frac{3 \cdot 4}{2} = 18$ , while the optimal solution for PCMSCP is  $\Lambda_2 = \{\{1, 3, 4, 6, 7, 9\}, \{2\}, \{5\}, \{8\}\}$ , having four cliques, and a chromatic sum equal to  $\frac{6 \cdot 7}{2} + 3 \cdot \frac{1 \cdot 2}{2} = 24$ .

**Definition 3.4** Let  $G = (V, E)$  be a graph. Partition into Cliques for MSCP (PCMSCP) consists in finding a partition of  $V$  into disjoint sets  $V_1, V_2, \dots, V_l$  such that for  $1 \leq i \leq l$ , the subgraph induced by  $V_i$  is a clique and  $\sum_1^l \frac{|V_i|(|V_i|+1)}{2}$  is maximum. The optimal solution of PCMSCP is denoted  $\sum_{clique}(G)$ .

Obviously,  $\{\{1\}\{2\}, \dots, \{|V|\}\}$  is a trivial partition of  $G = (V, E)$ , and in this case  $\sum_{clique}(G) \geq n$ . Graphs where  $\omega(G) = 2$  is another trivial case, for which PCMSCP amounts to solving the maximum matching problem, and can therefore be solved in polynomial time. Next, we establish a general upper bound of  $\sum_{clique}(G)$  according to  $\omega(G)$ .

**Proposition 3.5** Let  $\omega$  be the size of a maximum clique of  $G$ ,  $r = n - \omega \lfloor \frac{n}{\omega} \rfloor$ .  $\sum_{clique}(G)$  is upper bounded by  $MAX_{LBclique} = \frac{\omega(\omega+1)}{2} \lfloor \frac{n}{\omega} \rfloor + \frac{r(r+1)}{2}$ .

**Proof.** If  $G$  has a maximum clique of size  $\omega$ , any partition  $\Lambda$  has at most  $\lfloor \frac{n}{\omega} \rfloor$  cliques of size  $\omega$ . Each of these has an associated sum equal to  $\frac{\omega(\omega+1)}{2}$ . The  $r$  remaining vertices in the best case form a clique with an associated sum equal to  $\frac{r(r+1)}{2}$ .

Now let  $\Lambda_{opt} = V_1, V_2, \dots, V_l$  be an optimal solution of PCMSCP. Without

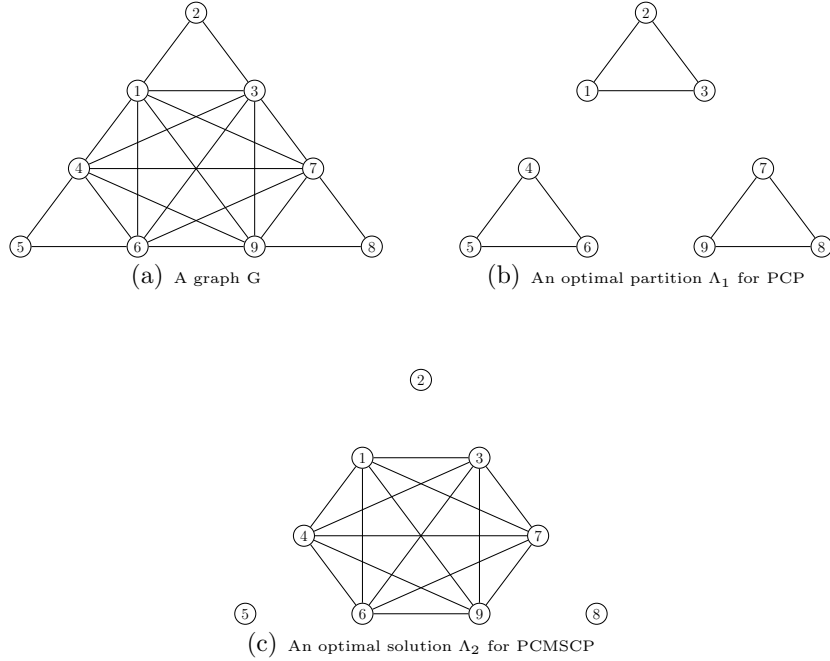


Fig. 4. Partition into Cliques for MSCP (PCMSCP) and Partition into Cliques (PCP)

loss of generality, we can assume that  $|V_1| \geq |V_2| \geq \dots \geq |V_l|$ . We construct a new partition of  $V$  into subsets  $X_1, X_2, \dots, X_p$  ( $p \leq l$ ), each subset  $X_i$  being initialized with  $V_i$ . So, initially, the vertices in  $X_i$  are colored from 1 to  $|V_i|$ . Consider the first subset  $X_j$ , ( $1 \leq j < p$ ) such that  $|X_j| < \omega$ . Remove from  $X_p$  vertex  $u$  of color  $|V_p|$  and add  $u$  to  $X_j$ . The new color of  $u$  will be  $|V_j| + 1 > |V_p|$ . Note that the subsets  $X_i$  are not necessarily cliques in  $G$  and  $\sum_1^l \frac{|V_i|(|V_i|+1)}{2} < \sum_1^p \frac{|X_i|(|X_i|+1)}{2}$ . By continuing in the same manner, we obtain a series of partitions, until we end up with the final partition, each of whose subsets has  $\omega$  vertices, with the exception of the last subset, which has  $r$  vertices. The result consequently holds.  $\square$

**corollary 1** *Let  $G = (V, E)$  be a graph such that  $|V|$  is a multiple of  $\omega(G)$ . If  $G$  is partitioned into  $k$  cliques  $V_1, \dots, V_k$  such as  $\sum_1^k \frac{|V_i|(|V_i|+1)}{2} \geq \frac{\omega(\omega+1)}{2} \frac{|V|}{\omega}$ , then  $|V_i| = \omega \forall i$  and  $\sum_1^k \frac{|V_i|(|V_i|+1)}{2} = \frac{\omega(\omega+1)}{2} \frac{|V|}{\omega}$ .*

**Theorem 3.6** *PCMSCP is an NP-hard problem.*

Given Corollary 1, and using the same local replacement from Exact Cover

by 3-Sets (X3C) to Partition Into Triangle (PIT) in [4], PCMSCP is shown to be NP-Hard (see Appendix).

The two lower bounds  $\sum_{tree}(G)$  and  $\sum_{clique}(G)$  are incomparable [DAVID : 'incomparable' .. que voulez-vous dire ?] (see the examples given in Figure 2 and Figure 3(a)). Also, note that the chromatic sum can be smaller than  $MAX_{LBbipart}$  and  $MAX_{LBclique}$ .

## 4 Solution strategy

### 4.1 General outline

MSCP is NP-hard, so we propose to compute lower and upper bounds. As discussed previously, an upper bound can be obtained by coloring the original graph  $G$ , while a lower bound is obtained by coloring a partial graph of  $G$ , and such a partial graph can be generated by partitioning  $G$  into cliques. Moreover, the problem of finding such a partition can easily be reduced to that of coloring the complementary graph of  $G$ , given the relation between a clique and an independent set.

Let  $\overline{G} = (V, E')$  be the complementary graph of  $G$  such that  $\forall (u, v) \in V \times V, [u, v] \in E'$  if and only if  $[u, v] \notin E$ . So, by definition, any independent set of  $\overline{G}$  is a clique of  $G$ . Therefore, any partition of  $\overline{G}$  into independent sets  $X_1, X_2, \dots, X_p$ , can be seen as a partition of  $G$  into  $p$  cliques. Now, finding a partition of  $\overline{G}$  into independent sets is simply equivalent to coloring  $\overline{G}$ .

The quality of the upper and lower bounds depends on the methods used for coloring  $G$  and  $\overline{G}$  respectively. We have chosen to use a single coloring heuristic algorithm to compute a lower bound, denoted  $LB_{clique}$  with the objective function *Maximise*  $f_{PCMSCP}(S) = \sum_{i=1}^p \frac{x_i \cdot (x_i + 1)}{2}$  for PCMSCP, and an upper bound denoted  $UB_{clique}$  with *Minimise*  $f_{MSCP}(S) = \sum_{i=1}^p i \cdot x_i$  for MSCP. This algorithm is known as the Memetic Algorithm for MSCP (MA-MSCP).

Memetic algorithms are recent metaheuristics [17] that belong to the family of Evolutionary Algorithms. A memetic algorithm consists of two phases of evolution: a genetic evolution phase and a phase of local evolution. In the phase of genetic evolution a memetic algorithm causes a population of individuals to evolve according to the genetic principle, while in the local phase of evolution individuals evolve according to the principle of local improvement. A memetic algorithm can thus be seen as a genetic algorithm hybridized

with local search techniques. It comprises a number of basic components: a representation of the solution, an initial population, the selection of individuals, the combination of individuals (crossover), local improvement and update of the population. We shall now describe in some detail how each of these components contributes to our approach for solving the MA-MSCP problem. Concerning PCMSCP, it is sufficient to consider its objective function  $f_{MSCP}(S) = \sum_{i=1}^p \frac{x_i \cdot (x_i + 1)}{2}$  in different components.

#### 4.2 Representation and evaluation of the solution

Each individual  $S$  is a coloring encoded as a partition  $X_1, \dots, X_k$  of the vertex set of the graph, such that  $x_1 \geq x_2 \geq \dots \geq x_k$ . The quality of  $S$  is estimated by its sum of colors  $f_{MSCP}(S) = \sum_{i=1}^k i \cdot x_i$ .

#### 4.3 Initial population

The population  $POP$  is a list of  $P$  individuals  $\{S_1, \dots, S_P\}$ , sorted in ascending order of  $f_{MSCP}$ . A good initial population is composed of individuals of good quality and diversity. In this context, we initialize our population with  $\frac{1}{4}P$  individuals generated by greedy algorithms proposed in [13] and  $\frac{3}{4}P$  individuals generated randomly. To maintain the diversity of this population, we do not allow two different individuals to have the same evaluation value, i.e. for  $S_i$  and  $S_j$ ,  $f_{MSCP}(S_i) \neq f_{MSCP}(S_j)$ . The population in our memetic algorithm size is fixed to 20 individuals.

#### 4.4 Selection of parents

To select parents we use a binary tournament selection, which proves better than a random selection or roulette-wheel selection. First, we randomly select four different individuals out of the population. Then from among the first two we choose the better individual, in terms of  $f_{MSCP}$ , to be parent  $P_1$ , and from among the remaining two the better individual to be the second parent  $P_2$ .

#### 4.5 Crossover

The crossover operator used in our MA-MSCP to produce a new individual  $S_{new}$  (offspring), is an adaptive GPX crossover operator proposed by Galinier and Hao [3]. Given two parents  $P_1$  with  $k_1$  colors and  $P_2$  with  $k_2$  colors, our crossover begins by building the  $k'$  ( $k' = \min\{k_1, k_2\}$ ) color classes of

$S_{new}$  as follows. At step  $l$  ( $1 \leq l \leq k'$ ) we build the color class  $X_l$  of  $S_{new}$  as follows: we consider parent  $P_1$  if  $l$  is odd, otherwise we consider parent  $P_2$ . In the considered parent, we choose the largest color class to become class  $X_l$  in  $S_{new}$ , and remove all its vertices from  $P_1$  and  $P_2$ . At the end of these steps, some vertices may remain unassigned. These vertices are then randomly assigned to an available color class. If no available color class exists, the vertices are assigned to a new color class.

#### 4.6 Local improvement

The purpose of the local search in MA-MSCP is to improve a new individual produced by the above crossover, before inserting it into the population. Different neighborhoods are tested in an exploration of the solution space. Among all the neighborhoods tested, the two presented here give the best solutions for MSCP as well as PCMSCP.

**Hill climbing:** a vertex  $v$  colored with  $c(v)$  is randomly chosen in the individual  $S$ , extracted from its color class  $X_{c(v)}$  and then inserted, if possible, into a color class  $X_p$  such that  $x_{c(v)} \leq x_p$  and  $p$  is an available color for  $v$ . We then obtain a new individual  $S'$ , and obviously  $f_{MSCP}(S') \leq f_{MSCP}(S)$ . This process is iterated  $n$  times.

**Destroy and repair:** the general principle is to remove a small part of the individual and improve it by reconstruction [18]. We apply this procedure to the individual obtained using the hill climbing method outlined above. Let  $S$ , be an individual composed of  $k$  color classes and  $d$  a random value between 1 and  $n/k$ . First, randomly remove  $d$  vertices  $v_1, v_2, \dots, v_d$  from  $S$ . Next, the solution is reconstructed such that each removed vertex  $v_i$  is inserted into the largest available color class  $X_p$  ( $1 \leq p \leq k$ ). If such a color class does not exist, the vertex  $v_i$  is assigned to a new color class, and the number of colors is incremented.

The two procedures are applied alternately until there is no further improvement. This schema defines our MA-MSCP mutation operator which is systematically applied to each new individual obtained by the crossover operator.

##### 4.6.1 Update population

To build the new population from the previous one, we proceed as follows. After the crossover and the local improvement, we obtain a new individual  $S_{new}$  characterized by  $f_{MSCP}(S_{new})$ . If there is an individual  $S$  in the population such that  $f_{MSCP}(S_{new}) = f_{MSCP}(S)$  then  $S$  is replaced by  $S_{new}$  to form the

next population. Otherwise,  $S_{new}$  is inserted according to its rank, and the worst individual of the next population is rejected.

The stopping criterion of our MA-MSCP algorithm is time. This algorithm will be executed for a maximum time  $T = 3$  hours, and its result is the best individual in the last population. Our memetic algorithm can be summarized in Algorithm 1.

**Algorithm 1.** Memetic Algorithm for MSCP

**Require:**  $POP$  (set of individuals  $\{S_1, \dots, S_P\}$ )

**Ensure:**  $S_1$  (best solution)

- 1: Initialize the population  $POP$  with  $P$  individuals
- 2: Sort individuals of  $POP$  in ascending order of  $f_{MSCP}$
- 3: **while** ( $T$  is not over) **do**
- 4:   Choose two parents  $P_1$  et  $P_2$  by binary tournament
- 5:    $S_{new} \leftarrow Crossover(P_1, P_2)$
- 6:    $S_{new} \leftarrow Local\ Search(S_{new})$
- 7:   **if** ( $\exists S, f_{MSCP}(S_{new}) = f_{MSCP}(S)$ ) **then**
- 8:     Replace  $S$  by  $S_{new}$
- 9:   **else**
- 10:    **if** ( $f_{MSCP}(S_{new}) \leq f_{MSCP}(S_P)$ ) **then**
- 11:     Insert  $S_{new}$  into  $POP$
- 12:     Reject  $S_P$  from  $POP$
- 13:    **end if**
- 14:   **end if**
- 15: **end while**
- 16: **return**  $S_1$

## 5 Experimental Results

In this section we present our experimental results for lower and upper bounds of MSCP obtained using our memetic algorithm. These bounds provide an interval within which the optimal solution is to be found for the instances presented. We compared  $LB_{clique}$  and  $UB$  for the instances used with some theoretical lower and upper bounds [20]. We also, compare our results of sum coloring with the best one in literature presented by Wu and al [21]. In their paper Wu et al. [21] propose a heuristic EXSCOL based on the extraction of independent sets. Each of these sets is a color class. The

idea developed is as follows: initially the size  $k$  of an independent set as large as possible in the initial graph  $G$  is calculated. Then, the algorithm searches the maximum number of independent sets, not necessarily disjoint, of size  $k$  in  $G$ . From these independent sets, EXSCOL constructs a graph  $G'$  representing the dependencies between them (common vertices). Finally, find an independent set as large as possible in  $G'$ , is to seek the greatest possible number of disjoint independent sets of size  $k$  in  $G$  by an approximate approach. These sets are removed from  $G$  and the principle is iterated until  $G$  is empty. The authors compare the results of the sum of EXSCOL dedicated to color a graph, with those of their memetic algorithm MACOL dedicated to the classic graph coloring, and thus show the difference in approach to solving both problems of color.

Our algorithm was programmed in C and run on an Intel Core 2 Duo T5450-1.66-1.67 with 2GB Ram running under Windows Vista Home Premium.

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$L_{b_{th}}$	$UB_{b_{th}}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
$dsjc125.1$	125	736	0, 09	4	5	187	?	135	375	247	35	247	326	6	10	326	7	1	0	326	0
$dsjc125.5$	125	3891	0, 50	10	17	187	?	177	1125	549	9	549	1013	18	5	1017	20	1	-4	1013	0
$dsjc125.9$	125	6961	0, 89	34	44	187	?	686	2812	1689	20	1689	2503	44	2	2512	44	1	-9	2503	0
$dsjc250.1$	250	3218	0, 10	4	8	375	?	256	1125	569	34	569	983	9	42	985	10	4	-2	983	0
$dsjc250.5$	250	15668	0, 50	12	28	375	?	355	3625	1280	40	1280	3214	30	26	3246	31	6	-32	3214	0
$dsjc250.9$	250	27897	0, 89	43	72	375	?	1153	9125	4279	16	4279	8277	73	33	8286	75	7	-9	8277	0
$dsjc500.1$	500	12458	0, 10	5	12	750	?	510	3250	1241	23	1241	2897	15	4	2850	14	9	47	2850	0
$dsjc500.5$	500	62624	0, 50	13	48	750	?	708	12250	2868	51	2868	11082	51	42	10910	51	11	172	10910	0
$dsjc500.9$	500	112437	0, 90	56	126	750	?	2040	31750	10759	17	10759	29995	132	51	29912	132	15	83	29912	0
$dsjc1000.1$	1000	49629	0, 10	6	21	1500	?	1015	11000	2707	99	2707	9188	23	31	9003	22	28	185	9003	0

following in next page ...



continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$Lb_{th}$	$Ub_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
$dsjc1000.5$	1000	249826	0, 50	15	87	1500	?	1414	44000	6534	35	6534	38421	90	23	37598	87	24	<b>823</b>	37598	<b>0</b>
$dsjc1000.9$	1000	449449	0, 90	67	224	1500	?	3211	112500	26157	61	26157	105234	235	61	103464	231	27	<b>1770</b>	103464	<b>0</b>
$dsjr500.1$	500	3555	0, 03		<b>12</b>	750	?	566	3250	2061	22	2061	2173	12	4					2173	<b>0</b>
$dsjr500.1c$	500	121275	0, 97	83	<b>84</b>	750	?	3986	21250	15025	12	15025	16311	92	34					16311	<b>0</b>
$dsjr500.5$	500	58862	0, 47	<b>122</b>	<b>122</b>	750	30090	7881	30750	22728	31	22728	25630	136	44					25630	<b>0</b>
$flat300 - 20 - 0$	300	21375	0, 48	11	<b>20</b>	450	?	490	3150	1515	59	1515	3150	20	1	3150	20	3	<b>0</b>	3150	<b>0</b>
$flat300 - 26 - 0$	300	21633	0, 48	11	<b>26</b>	450	?	625	4050	1536	42	1536	3966	26	1	3966	26	3	<b>0</b>	3966	<b>0</b>
$flat300 - 28 - 0$	300	21695	0, 48	12	<b>28</b>	450	?	678	4350	1541	27	1541	4261	32	28	4282	34	3	<b>-21</b>	4261	<b>0</b>
$flat1000 - 50 - 0$	1000	245000	0, 49	14	<b>50</b>	1500	?	2225	25500	6433	64	6433	25500	50	28	25500	50	9	<b>0</b>	25500	<b>0</b>
$flat1000 - 60 - 0$	1000	245830	0, 49	15	<b>60</b>	1500	?	2770	30500	6402	43	6402	30100	60	16	30100	60	11	<b>0</b>	30100	<b>0</b>
$flat1000 - 76 - 0$	1000	246708	0, 49	15	<b>76</b>	1500	?	3850	38500	6330	67	6330	38213	90	8	37167	86	19	<b>1046</b>	37167	<b>0</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$Lb_{th}$	$UB_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	<b>Diff</b>	$UB$	<b>Opt</b>
$fpsol2.i.1$	496	11654	0, 09	<b>65</b>	<b>65</b>	744	15876	2576	12150	3403	64	3403	3403	65	4					3403	<b>1</b>
$fpsol2.i.2$	451	8691	0, 09	<b>30</b>	<b>30</b>	676	6976	886	6990	1668	29	1668	1668	30	3					1668	<b>1</b>
$fpsol2.i.3$	425	8688	0, 10	<b>30</b>	<b>30</b>	637	6525	860	6587	1636	19	1636	1636	30	2					1636	<b>1</b>
$le450 - 5a$	450	5714	0, 06	<b>5</b>	<b>5</b>	675	1350	460	1350	1190	30	1190	1351	6	1					1350	<b>0</b>
$le450 - 5b$	450	5734	0, 06	<b>5</b>	<b>5</b>	675	1350	460	1350	1186	16	1186	1352	6	2					1350	<b>0</b>
$le450 - 5c$	450	9803	0, 10	<b>5</b>	<b>5</b>	675	1350	460	1350	1272	42	1272	1351	6	3					1350	<b>0</b>
$le450 - 5d$	450	9757	0, 10	<b>5</b>	<b>5</b>	675	1350	460	1350	1269	46	1269	1351	6	2					1350	<b>0</b>
$le450 - 15a$	450	8168	0, 08	<b>15</b>	<b>15</b>	675	3600	555	3600	2329	4	2329	2681	18	19	2632	18	5	49	2632	<b>0</b>
$le450 - 15b$	450	8169	0, 08	<b>15</b>	<b>15</b>	675	3600	555	3600	2348	3	2348	2690	19	19	2642	19	7	48	2642	<b>0</b>
$le450 - 15c$	450	16680	0, 16	<b>15</b>	<b>15</b>	675	3600	555	3600	2585	7	2585	3943	24	6	3866	24	6	77	3600	<b>0</b>
$le450 - 15d$	450	16750	0, 17	<b>15</b>	<b>15</b>	675	3600	555	3600	2622	25	2622	3926	24	3	3921	26	5	5	3600	<b>0</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$Lb_{th}$	$UB_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
$le450 - 25a$	450	8260	0, 08	<b>25</b>	<b>25</b>	675	5850	750	5850	3003	6	3003	3178	28	5	3153	26	7	<b>25</b>	3153	<b>0</b>
$le450 - 25b$	450	8263	0, 08	<b>25</b>	<b>25</b>	675	5850	750	5850	3305	3	3305	3379	26	7	3366	26	6	<b>13</b>	3366	<b>0</b>
$le450 - 25c$	450	17343	0, 17	<b>25</b>	<b>25</b>	675	5850	750	5850	3627	32	3627	4648	33	16	4515	31	8	<b>133</b>	4515	<b>0</b>
$le450 - 25d$	450	17425	0, 17	<b>25</b>	<b>25</b>	675	5850	750	5850	3697	21	3697	4696	31	3	4544	31	7	<b>152</b>	4544	<b>0</b>
$multisol.i.1$	197	3925	0, 20	<b>49</b>	<b>49</b>	295	4901	1373	4122	1957	5	1957	1957	49	2					1957	<b>1</b>
$multisol.2$	188	3885	0, 22	<b>31</b>	<b>31</b>	282	2979	653	3008	1191	6	1191	1191	31	2					1191	<b>1</b>
$multisol.i.3$	184	3916	0, 23	<b>31</b>	<b>31</b>	276	2915	649	2944	1187	7	1187	1187	31	1					1187	<b>1</b>
$multisol.i.4$	185	3946	0, 23	<b>31</b>	<b>31</b>	277	2945	650	2960	1189	5	1189	1189	31	1					1189	<b>1</b>
$multisol.i.5$	186	3973	0, 23	<b>31</b>	<b>31</b>	279	2976	651	2976	1160	3	1160	1160	31	1					1160	<b>1</b>
$inithx.i.1$	864	18707	0, 05	<b>54</b>	<b>54</b>	1296	23760	2295	19571	3676	121	3676	3676	54	5					3676	<b>1</b>
$inithx.i.2$	645	13979	0, 07	<b>31</b>	<b>31</b>	967	10245	1110	10320	2050	17	2050	2050	31	10					2050	<b>1</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$Lb_{th}$	$Ub_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	<b>Diff</b>	$UB$	<b>Opt</b>
<i>mitha.i.3</i>	621	13969	0, 07	<b>31</b>	<b>31</b>	931	9921	1086	9936	1986	90	1986	1986	31	2					1986	<b>1</b>
<i>zeroin.i.1</i>	211	4100	0, 18	<b>49</b>	<b>49</b>	316	5020	1387	4311	1822	10	1822	1822	49	2					1822	<b>1</b>
<i>zeroin.i.2</i>	211	3541	0, 16	<b>30</b>	<b>30</b>	316	3256	646	3270	1004	14	1004	1004	30	1					1004	<b>1</b>
<i>zeroin.i.3</i>	206	3540	0, 17	<b>30</b>	<b>30</b>	309	3141	641	3193	998	37	998	998	30	1					998	<b>1</b>
<i>queen5 - 5</i>	25	160	0, 51	<b>5</b>	<b>5</b>	37	75	36	75	75	1	75	75	5	1		5	1	<b>0</b>	75	<b>1</b>
<i>queen6 - 6</i>	36	290	0, 45	<b>6</b>	<b>7</b>	54	126	57	144	126	1	126	138	7	1		10	1	<b>-12</b>	138	<b>0</b>
<i>queen7 - 7</i>	49	476	0, 40	<b>7</b>	<b>7</b>	73	196	70	196	196	1	196	196	7	1		7	1	<b>0</b>	196	<b>1</b>
<i>queen8 - 8</i>	64	728	0, 36	<b>8</b>	<b>9</b>	96	288	100	320	288	1	288	291	9	1		9	1	<b>0</b>	291	<b>0</b>
<i>queen8 - 12</i>	96	1368	0, 30	<b>12</b>	<b>12</b>	144	624	162	624	624	1	624	624		1					624	<b>1</b>
<i>queen9 - 9</i>	81	1056	0, 32	<b>9</b>	<b>10</b>	121	405	126	445	405	1	405	409	10	1					409	<b>0</b>
<i>queen10 - 10</i>	100	1470	0, 29	<b>10</b>	<b>11</b>	150	550	155	600	550	1	550	553	11	2					553	<b>0</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Max_b$	$Max_c$	$Lb_{th}$	$UB_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
<i>queen11 – 11</i>	121	1980	0, 27	<b>11</b>	<b>11</b>	181	726	176	726	726	1	726	733	12	6					726	<b>1</b>
<i>queen12 – 12</i>	144	2596	0, 25	<b>12</b>	<b>12</b>	216	936	210	936	936	1	936	944	13	14					936	<b>1</b>
<i>queen13 – 13</i>	169	3328	0, 23	<b>13</b>	<b>13</b>	253	1183	247	1183	1183	1	1183	1192	14	6					1183	<b>1</b>
<i>queen14 – 14</i>	196	4186	0, 22	<b>14</b>	<b>14</b>	294	1470	287	1470	1470	1	1470	1482	15	24					1470	<b>1</b>
<i>queen15 – 15</i>	225	5180	0, 20	<b>15</b>	<b>15</b>	337	1800	330	1800	1800	2	1800	1814	17	21					1800	<b>1</b>
<i>queen16 – 16</i>	256	6320	0, 19	<b>16</b>	<b>16</b>	384	2176	376	2176	2176	2	2176	2197	17	32					2176	<b>1</b>
<i>school1.col</i>	385	19095	0, 26	<b>14</b>	<b>14</b>	577	2863	476	2887	2345	6	2345	2674	15	25					2674	<b>0</b>
<i>school1 – nsh.col</i>	352	14612	0, 24	<b>14</b>	<b>14</b>	528	2628	443	2640	2106	42	2106	2392	15	8					2392	<b>0</b>
<i>anna</i>	138	493	0, 05	<b>11</b>	<b>11</b>	207	813	193	631	273	2	273	276	11	1	283	11	2	-7	276	<b>0</b>
<i>david</i>	87	406	0, 11	<b>11</b>	<b>11</b>	130	517	142	493	234	1	234	237	11	1	237	11	1	0	237	<b>0</b>
<i>Homer</i>	561	1628	0, 01	<b>13</b>	<b>13</b>	841	3916	639	2189	1129	52	1129	1157	13	32					1157	<b>0</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$K_i$	$Ma_{av}$	$Max_c$	$Lb_{th}$	$Ub_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
<i>huck</i>	74	301	0, 11	<b>11</b>	<b>11</b>	111	432	129	375	243	1	243	243	11	1	243	11	1	<b>0</b>	243	<b>1</b>
<i>jean</i>	80	254	0, 08	<b>10</b>	<b>10</b>	120	440	125	334	216	1	216	217	10	1	217	10	1	<b>0</b>	217	<b>0</b>
<i>games120</i>	120	638	0, 09	<b>9</b>	<b>9</b>	180	591	156	600	442	1	442	443	9	1	443	9	2	<b>0</b>	443	<b>0</b>
<i>miles250</i>	128	387	0, 05	<b>8</b>	<b>8</b>	192	576	156	515	318	1	318	325	8	8	328	9	2	<b>-3</b>	325	<b>0</b>
<i>miles500</i>	128	1170	0, 14	<b>20</b>	<b>20</b>	192	1296	318	1298	686	1	686	708	20	12	709	20	2	<b>-1</b>	708	<b>0</b>
<i>miles750</i>	128	2113	0, 26	<b>31</b>	<b>31</b>	192	1994	593	2048	1145	1	1145	1173	31	11					1173	<b>0</b>
<i>miles1000</i>	128	3216	0, 39	<b>42</b>	<b>42</b>	192	2712	989	2752	1623	13	1623	1679	42	5					1679	<b>0</b>
<i>miles1500</i>	128	5198	0, 63	<b>73</b>	<b>73</b>	192	4241	2756	4736	3239	1	3239	3354	73	1					3354	<b>0</b>
<i>myciel3</i>	11	20	0, 33	<b>2</b>	<b>4</b>	16	16	17	27	16	1	17	21	4	1	21	4	1	<b>0</b>	21	<b>0</b>
<i>myciel4</i>	23	71	0, 27	<b>2</b>	<b>5</b>	34	34	33	69	34	1	34	45	5	1	45	5	1	<b>0</b>	45	<b>0</b>
<i>myciel5</i>	47	236	0, 21	<b>2</b>	<b>6</b>	70	70	62	164	70	1	70	93	6	1	93	6	1	<b>0</b>	93	<b>0</b>

following in next page ...

continued from previous page

$G(V, E)$	$n$	$m$	$d$	$w$	$Ki$	$Max_b$	$Max_c$	$Lb_{th}$	$UB_{th}$	$LB_{MA}$	$CPU_{LB}$	$LB$	$UB_{MA}$	$k_{MA}$	$CPU_{UB}$	$UB_{JH}$	$k_{JH}$	$CPU_{JH}$	Diff	$UB$	Opt
<i>myciel6</i>	95	755	0, 17	<b>2</b>	<b>7</b>	142	142	116	380	142	1	142	189	7	1	189	7	2	0	189	0
<i>myciel7</i>	191	2360	0, 13	<b>2</b>	<b>8</b>	286	286	219	859	286	1	286	381	8	1	381	8	2	0	381	0
<i>latin - square-10.col</i>	900	307350	0, 76			1350	?	1569	450	40950	1	40950	41868	107	157	42223	109	4	-355	450	0
<i>C2000.5</i>	2000	999836	0, 50			3000	?	2829	1000	14382	4	14382	141213	164	168	132515	150	656	8698	1000	0
<i>C4000.5</i>	4000	4000268	0, 50			6000	?	5658	2000	31261	38	31261	522961	300	145	473234	266	2588	49727	2000	0
<i>wap05a.col</i>	905	43081	0, 11			1357	?	905	452	12365	1	12365	14709	52	147	13680	51	21	1029	452	0
<i>wap06a.col</i>	947	43571	0, 10			1420	?	947	473	0	0	947	14051	50	9	13778	48	27	273	473	0
<i>wap07a.col</i>	1809	103368	0, 06			2713	?	1809	904	24234	4	24234	29589	53	75	28629	51	112	960	904	0
<i>wap08a.col</i>	1870	104176	0, 06			2805	?	1870	935	24830	5	24830	31234	52	1	28896	51	127	2338	935	0
<i>qq.order-30.col</i>	900	26100	0, 06	<b>30</b>	<b>30</b>	1350	13950	1335	13950	13950	1	13950	13950	30	1	13950	30	28	0	13950	1
<i>qq.order-40.col</i>	1600	62400	0, 05			2400	?	2380	32800	32800	2	32800	32800	40	1	32800	40	35	0	32800	1

following in next page ...





Table 5 sums up the experimental results on instances from the DIMACS Library [6]. For each instance in Table 5, we denote as  $\omega^*$  the best lower bound for the maximum clique problem, as  $k^*$  the best upper bound for GCP, as  $d$  the density of the graph, and as  $LB_{th} = \max\{\lceil \sqrt{8m} \rceil, n + \frac{\chi(G)(\chi(G)-1)}{2}\}$  the theoretical lower bound given in [20]. Note that when  $\chi(G)$  is not known, we use the best value obtained for the maximum clique problem (column  $\omega^*$ ) in order to evaluate  $LB_{th}$ . For  $LB_{clique}$ , we present the results obtained by our memetic algorithm. We denote as  $UB_{th} = \min\{\frac{n(k_{best}+1)}{2}, \lfloor \frac{3(m+1)}{2} \rfloor\}$  the theoretical upper bound [20], and when  $\chi(G)$  is not known we use  $k_{best}$ . For the results of the lower bound of MSCP we use three columns. In the first column,  $LB_{MA}$ , we give the results of the lower bound obtained by the memetic algorithm, in the second column,  $CPU_{LB}$  the corresponding cpu time, and in the third column  $LB$  we present the best value of the lower bound between theoretical lower bound and this obtained by memetic algorithm. For the results of the upper bound of MSCP we use also three columns. In the first column,  $UB_{MA}$ , we give the results of the upper bound obtained by the memetic algorithm, in the second column,  $k_{MA}$  the corresponding number of colors, and in the third column  $CPU_{MA}$  we present the corresponding cpu time. We present the result of literature [21] on three column. In column,  $UB_{JH}$ , we give the results of the upper bound obtained by EXSCOL algorithm, in the second column,  $k_{JH}$  the corresponding number of colors, and in the third column  $CPU_{JH}$  we present the corresponding cpu time. Note that, the cpu time is reported in minutes. The column denoted by *Diff* represents the gap between our memetic algorithm and EXSCOL algorithm. That, denoted by *UB* we present the best value of the upper bound between theoretical upper bound, this proposed by Wu and al. and this obtained by our memetic algorithm.

The results in Table 5 show that  $LB_{clique}$  improves the theoretical lower bound  $LB_{th}$ . Indeed, only the number of vertices, the number of edges and the chromatic number are considered in  $LB_{th}$ , while  $LB_{clique}$  uses our memetic algorithm MA dedicated to MSCP. The results of  $MAX_b$  are still dominated by those of the lower bound based on partitioning the graph into cliques. For this reason we do not report results for the lower bound based on partial bipartite graphs, but present only those based on partitioning the graph into cliques. The bound  $MAX_c$  allows to identify 19 instances among the 88 tested for which the lower bound cannot be improved by our method of partitioning the graph into cliques.

Concerning the upper bound, we compare our memetic algorithm with EXSCOL on 55 instances tested by Wu and al. and gives first results on others families of graphs, as *fpsol*, *initx.i*, *mulsol.i*, *zeroin.i*, and large graphs as (*dsjr*), *school1.col*, and *school1 - nsh.col*. Also we complete the results on *queen*, *le450*, and *miles* families.

on the 55 instances tested in [21], our memetic algorithm strictly improves EXSCOL on 12 instances, they are equal on 19 instances, and the results of EXSCOL is better than our MA on the 21 remaining instances. We note also, that our MA is more efficient than EXSCOL on COLOR02 graphs [6], it obtains a worse result in only 4 graphs among the 23 presented in [21].

Comparing the best lower bound and the best upper bound allows us to identify 27 instances for which the optimum is reached. The optimality was identified by our memetic algorithm for 21 instances among these 27, and by the theoretical upper bound on the 6 remaining. The algorithm *EXSCOL* reach optimality just for 5 instances, for wich the sum coloring is equal to the sum coloring obtained by our MA.

## 6 Conclusion

In this paper we investigated lower and upper bounds for MSCP. Because the chromatic sum of any partial graph is a lower bound for MSCP, we considered a number of partial graphs such as spanning trees, partitions into paths and partitions into cliques, for which the chromatic sum is easy to compute. In this context we proposed a new NP-hard problem, Partition into Cliques for MSCP (PCMSCP), such that any solution of PCMSCP is a lower bound for MSCP. Building this kind of solution is simply a matter of coloring the complementary graph in order to work out a partition of the graph into cliques. We then presented a single coloring algorithm MA-MSCP, with different objective functions, but a similar procedure, for computing the lower and upper bounds. MA-MSCP is a memetic algorithm, based on an adaptive GPX crossover operator and on a local search dedicated to MSCP. Experimental results for lower and upper bounds show that our approach improves the results for most of the literature instances and gives the optimum for 26 instances out of 76.

## References

- [1] A. Bar-Noy and G. Kortsarz. Minimum color sum of bipartite graphs. *J. Algorithms*, 28(2):339–365, 1998.
- [2] D. Brasz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.
- [3] P. Galinier and J.K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.
- [4] R. Garey and S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. 1990.
- [5] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.
- [6] <http://mat.gsia.cmu.edu/COLOR/instances.html>.
- [7] K. Jansen. The optimum cost chromatic partition problem. In *CIAC*, pages 25–36, 1997.
- [8] K. Kokosiński and K. Kwarciany. On sum coloring of graphs with parallel genetic algorithms. In *ICANNGA '07*, pages 211–219, 2007.
- [9] L.G. Kroon, A. Sen, H. Deng, and A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. In *WG '96*, pages 279–292, 1997.
- [10] E. Kubicka and A.J. Schwenk. An introduction to chromatic sums. In *CSC '89: Proceedings of the 17th conference on ACM Annual Computer Science Conference*, pages 39–45, 1989.
- [11] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, Winston, 1976.
- [12] F.T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of research of the national institute of standards and technology*, 84(6):251–256, 1979.
- [13] Y. Li, C. Lucet, A. Moukrim, and K. Sghiouer. Greedy algorithms for the minimum sum coloring problem. In *International Workshop: Logistics and transport*, 2009.
- [14] C. Lucet, F. Mendes, and A. Moukrim. An exact method for graph coloring. *Computers & Operations Research*, 33(8):2189–2207, 2006.
- [15] M. Malafiejski. *Sum coloring of graphs, in Graph Colorings (M.Kubale, Ed.)*. 2004.

- [16] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.
- [17] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms, 1989.
- [18] R. Ruiz and T. Sttzle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2006.
- [19] M.R. Salavatipour. On sum coloring of graphs. *Discrete Appl. Math.*, 127(3):477–488, 2003.
- [20] C. Thomassen, P. Erds, Y. Alavi, P.J. Malde, and A.J. Schwenk. Tight bounds on the chromatic sum of a connected graph. *J. Graph Theory*, 13:353–357, 1989.
- [21] Qinghua Wu and Jin-Kao Hao. An effective heuristic algorithm for sum coloring of graphs. *Comput. Oper. Res.*

## 7 Appendix

First we recall the two problems.

- Exact Cover by 3-Sets (X3C):  
 Instance: a finite set  $X$  with  $|X| = 3k$  and a collection  $U$  of 3-element subsets of  $X$ .  
 Question: Does  $U$  contain an exact cover of  $X$ , that is, a subcollection  $U' \subseteq U$  such that every element of  $X$  occurs in exactly one member of  $U'$ ?
- Partition into cliques for MSCP (PCMSCP):  
 Instance: let  $G = (V, E)$  be a graph with  $|V| = 3k$  for a positive integer  $k$ , and  $B$  an integer.  
 Question: Is there a partition of  $V$  into  $k$  disjoint subsets  $V_1, V_2, \dots, V_k$  of  $V$  such that  $\forall i$   $1 \leq i \leq k$ , the subgraph induced by  $V_i$  is a complete graph and  $\sum_1^k \frac{|V_i|(|V_i|+1)}{2} \geq B$ ?

It is easy to see that  $\text{PCMSCP} \in \text{NP}$ . We shall show that the Exact Cover by 3-Sets problem can be transformed into Partition into cliques for MSCP.

Let the set  $X$  with  $|X| = 3k$  and the collection  $U = \{U_1, \dots, U_s\}$  be an arbitrary instance of X3C.

We shall construct an instance of PCMSCP as follows: for each  $i$  in  $\{1, \dots, s\}$  and  $U_i = \{x_i, y_i, z_i\}$  in  $U$ , we introduce 18 edges as in Figure [4]. These edges will be denoted by  $E_i$ . We introduce a graph  $G = (V, E)$  where  $V = X \cup \bigcup_1^s \{a_i(j), 1 \leq j \leq 9\}$ ,  $E = \bigcup_1^s E_i$  and  $B = 2|V|$ . Note that the graph  $G$  and the number  $B$  can easily be obtained from X3C in polynomial time.

Let  $\{U_{i_1}, \dots, U_{i_q}\}$  be [DAVID : j'ai remplac 'of' par 'be', pour que la phrase soit grammaticalement correcte, mais j'ai un doute concernant le sens .. vrifier] subsets in any exact cover for  $X$ . If  $i \in \{1, \dots, s\}$ , there are two possible cases:

**Case 1:**  $U_i \in \{U_{i_1}, \dots, U_{i_q}\}$ . We can therefore construct 4 cliques of  $G$ :  $\{a_i(1), a_i(2), x_i\}$ ,  $\{a_i(4), a_i(5), y_i\}$ ,  $\{a_i(7), a_i(8), z_i\}$ ,  $\{a_i(3), a_i(6), a_i(9)\}$ .

**Case 2:**  $U_i \notin \{U_{i_1}, \dots, U_{i_q}\}$ . We can therefore build 3 cliques of  $G$ :  $\{a_i(1), a_i(2), a_i(3)\}$ ,  $\{a_i(4), a_i(5), a_i(6)\}$ ,  $\{a_i(7), a_i(8), a_i(9)\}$ .

We obtain a partition  $Z = \{V_1, \dots, V_{\lfloor \frac{|V|}{3} \rfloor}\}$  of  $G$  into 3- cliques and  $\sum_1^{\lfloor \frac{|V|}{3} \rfloor} \frac{|V_i|(|V_i|+1)}{2} = \sum_1^{\lfloor \frac{|V|}{3} \rfloor} \frac{3(4)}{2} = 2|V|$ .

Conversely, assume that  $T = \{V_1, \dots, V_k\}$  is a partition of  $G$  into  $k$  cliques such that  $\sum_1^k \frac{|V_i|(|V_i|+1)}{2} \geq 2|V|$ .

Note that  $\omega(G) = 3$ , using Corollary 1, we have:  $\sum_1^k \frac{|V_i|(|V_i|+1)}{2} = 2|V|$  and  $|V_i| = 3 \forall i \in \{1, \dots, k\}$ . Consequently,  $\forall i \in \{1, \dots, s\}$ ,  $\{a_i(3), a_i(6), a_i(9)\} \in T$ . The subsets  $\{x_i, y_i, z_i\}$  therefore constitute an exact cover for  $X$ .

## A Memetic Algorithm for staff scheduling problem in airport security service

Anas Abdoul Soukour<sup>a,b</sup>, Laure Devendeville<sup>b\*</sup>, Corinne Lucet<sup>b</sup>  
and Aziz Moukrim<sup>c</sup>

<sup>a</sup>*DSI, ICTS France, Roissypole le dôme, 1 rue de la Haye, BP 12936, 95732  
Roissy CDG Cedex, France;*

<sup>b</sup>*Laboratoire de MIS - EA 4290 / Université de Picardie Jules Verne, 33 rue  
Saint Leu, 80039 Amiens Cedex 1, France;*

<sup>c</sup>*Laboratoire HeuDiaSyC - CNRS UMR 6599 / Université de Technologie de  
Compiègne, Centre de Recherches de Royallieu, BP 20529, 60205 Compiègne  
Cedex, France*

*(Received 00 Month 20xx; final version received 00 Month 20xx)*

The staff scheduling problem is widely studied in Operational Research. Various surveys are available in the literature dealing with this problem which concerns various objectives and various constraints. In this article, we present a staff scheduling problem in airport security service. First, a modeling of the problem, and a formalization of solutions are shown. The problem is solved in three steps, days-off scheduling, shift scheduling, and staff assignment. We focus on the last step, by providing a Memetic Algorithm (*MA*) which merged an Evolutionary Algorithm and Local Search techniques. We propose a chromosome encoding, a crossover operator and a combined neighborhood function, specially dedicated to this staff assignment problem. Besides providing better solutions than software currently used, this algorithm provides up to 50% of improvement from initial feasible solutions.

**Keywords:** planning, optimization problem, modelling, resolution, scheduling algorithms, metaheuristics,

### 1. Introduction

This paper presents a scheduling problem in airport security service. In this particular application area, human workforce is spread to control various check points on airport ground, for airlines and airport operators. Activities consist in controlling and/or supervising strategic points of a particular geographical area involving a large number of

---

\*Corresponding author. Email: laure.devendeville@u-picardie.fr

persons either moving (tarmac, baggage area, etc.) or remaining in a location (check-in, boarding area, etc.) for a certain time. Scheduling these activities, distributed mostly to large geographical areas, is done according to legal constraints, preferences of employees and customer's requirements. One of the reasons of the problem complexity is due to large quantity of employees working in different places. Employment contracts are various and numerous. Moreover they are specified with multiple legal regulations, due to french law, and collective agreements. In addition, there are many sundry kind of skills with different levels. Moreover, employee requirement expressed by airport operators or airlines is characterized by a large fluctuation in the horizon time.

Staff scheduling is a hard optimisation problem (Garey and Johnson (1979)). Kragelund and Kabel (1998) show the NP-hardness of the general employee timetabling problem. Besides, Tien and Kamiyama (1982) proves that practical personnel scheduling problems are generally more complex than the Travelling Salesman Problem which is itself NP-hard. Various surveys are available in literature dealing with scheduling problem (Blöchliger (2004), Ernst *et al.* (2004), Ásgeirsson (2010), Aparecido Constantino *et al.* (2010), Métivier *et al.* (2010), Bäumelt *et al.* (2010)).

Solving staff scheduling problem is generally decomposed in three or four steps, for complexity reasons (Boris *et al.* (2009)), but also in order to include data depending on feedback experiences. The first step, *days-off scheduling*, consists in specifying a sequence of days of work and days-off for each employee according to requirement of number of employees. It is constrained by employee's contract (for example working only weekends or specific days of week) or collective agreements (for example number of weekends off). The second step consists in selecting, from a potentially large pool of shifts, what shifts are to be worked, and how many are needed in order to meet demand. For this *shift scheduling* problem, two kinds of models exist in literature: task covering model, based on Dantzig model (Soumis *et al.* (2005)), implicit models defined by Rekik *et al.* (2004), Aykin (1998) or also Bechtold and Jacobs models (Bechtold and Jacobs (1990)). The last step, *staff assignment*, consists in assigning shifts, defined in step two, to employees, with the best satisfaction, and with respect of rules and regulations. Data from staff scheduling problem are various and numerous. Each combination of them provides a particular problem which is commonly solved, in industrial context, by heuristic methods or metaheuristics.

Our problem was decomposed into these three steps previously presented. In order to validate our model step by step we choose to solve first the last step, staff assignment, with data provided by the current system used by planning operators. In this paper we propose a metaheuristic based on a Memetic Algorithm. Memetic Algorithms (*MA*) are classified as Evolutionary Algorithms and are first introduced by Moscato (1999) and several studies showed its efficiency (Bouly *et al.* (2010)). It consists in a combination of an evolutionary process with local search (LS) methods to improve the population.

In section 2, we first show an integer programming modeling of the problem that allows to clearly express all the constraints. We also define and formalize solutions with their associated operators. The criteria we used to evaluate the quality of a solution are detailed. Section 3 develops our Memetic Algorithm approach for the assignment problem. An efficient chromosome encoding, a crossover operator and a combined neighborhood function are exposed. We also remind some greedy algorithms from previous works, used to initialize the population. Numerical results on real instances and their analysis are presented in section 4, then conclusions and perspectives end this article.

## 2. Problem modelling and construction of a feasible solution

For a given time period, the horizon, we have to satisfy demand expressed by a number of employees required in time unit that we call *workforce demand*. This demand is defined by clients and transformed by planning operators in a set of *shifts*. To fulfil this demand, the company has a fixed number of employees. To each employee is associated a plan or a cycle, specifying the workdays and the days off. Our aim is to assign the set of shifts to available employees during this period, optimizing a cost function.

### 2.1. Data

A *shift* represents a security activity of an employee, for a day of the horizon. It is defined by a starting and ending times, required skill and geographical area. It also includes meal break. Overall, shifts are defined according to working laws, especially controlling the respect of maximal and minimal legal hours. A shift may contain different tasks requiring different skills and possibly on different geographical areas. Travel time between two areas is considered when making shifts. In a general way, skills are hierarchically organized. When multi-skill is required, skill with upper level is selected as required skill for the considered shift. As an example, figure 1 shows a shift that starts at 8:00 and finishes at 16:00. Its associated skill is SKILL A, because of the higher level of SKILL A in the skill hierarchy and it is concerned by the two areas 1 & 2.

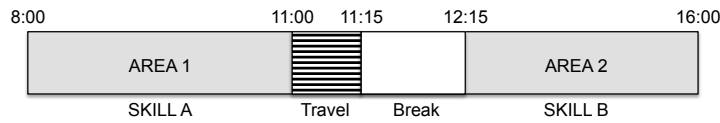


Figure 1. Shift example

In order to satisfy workforce demand, we have a set of *employees*. Each employee is characterized by a type of contract that fixes his monthly working time, a plan (cycle) of workdays and days off through the horizon, his higher level skill and some personal restrictions. In some case, medical restrictions prevent some employee from assignment on some geographical areas. Some employees work only on week-end and others never on Wednesday. Each employee is assigned a start time below which he can not start working and an end time beyond which he can not complete its work. An employee is *over time scheduled* if the sum of his assigned shifts during the horizon is greater than this contractual working time. He is *under time scheduled* otherwise.

Figure 2 shows an example of employee characteristics and his plan for an horizon of 30 days.

Employee X	SKILL		Type of Contract		Start time		End time		Restriction																						
	A		145 hours		8:00		22:00		no																						
j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
	off	off						off	off	off					off	off							off	off	off						

Figure 2. Plan example for an employee X



## 2.2. Modelisation

To provide a more formal explanation of the problem we have included an integer-programming modeling of our specific staff scheduling problem in this section. We present the main constraints on working laws and assignment consistency, and also the main criteria of cost evaluation.

### Data

$Q$ : set of shifts  $q \in Q$

$J$ : set of days  $j \in J$

$E$ : set of employees  $e \in E$

$C$ : set of skills  $c \in C$

$Z$ : set of areas  $z \in Z$

$\alpha_q^j = 1$  if shift  $q$  belongs to day  $j$

$\beta_j^e = 1$  if employee  $e$  is available day  $j$

$\gamma_q^c = 1$  if shift  $q$  requires skill  $c$

$\delta_e^c = 1$  if employee  $e$  has skill  $c$

$\theta_q^z = 1$  if shift  $q$  deals with area  $z$

$\zeta_e^z = 1$  if employee  $e$  can work on area  $z$

$\mu_e^j$ : the imperative start time of the employee  $e$  for day  $j$

$\rho_e^j$ : the imperative end time of the employee  $e$  for day  $j$

$\omega_q$ : starting time of shift  $q$

$\nu_q$ : ending time of shift  $q$

$l_q$ : duration of shift  $q$

$MH_e$ : number of contractual hours per month for employee  $e$

$Start_{week}$ : set of first day of weeks

$\$q$ : cost of the non assignment of the shift  $q$   $\$sup_e$ : cost of an extra working hour for employee  $e$

$\$inf_e$ : cost of missing working hour (under contract) for employee  $e$ .

### Variable

$x_q^e = 1$  if shift  $q$  is assigned to employee  $e$

### Minimize function:

$$\$UnderC + \$OverT + \$UnderT + \$EmpD \quad (1)$$

### Under constraints:

$$\forall e \in E, \forall q \in Q, \forall j \in J, \alpha_q^j(x_q^e - \beta_j^e) \leq 0 \quad (2)$$

$$\forall e \in E, \forall q \in Q, \forall c \in C, \gamma_q^c(x_q^e - \delta_e^c) \leq 0 \quad (3)$$

$$\forall e \in E, \forall q \in Q, \forall z \in Z, \theta_q^z(x_q^e - \zeta_e^z) \leq 0 \quad (4)$$

$$\forall e \in E, \forall j \in J, \sum_{q \in Q} \alpha_q^j \cdot x_q^e \leq 1 \quad (5)$$

$$\forall q \in Q, \sum_{e \in E} x_q^e \leq 1 \quad (6)$$

$$\forall e \in E, q \in Q, j \in J, (\mu_e^j - \omega_q) \cdot \alpha_q^j \cdot x_q^e \leq 0 \quad (7)$$

$$\forall e \in E, q \in Q, j \in J, (\nu_q - \rho_e^j) \cdot \alpha_q^j \cdot x_q^e \leq 0 \quad (8)$$

$$\forall e \in E, \forall j \in [1, J - 6], \sum_{j'=j}^{j+6} \sum_{q \in Q} \alpha_q^{j'} \cdot x_q^e \leq 6 \quad (9)$$

$$\forall e \in E, \forall j \in [1, J - 1], \left( \sum_{s \in S} \alpha_q^j \cdot x_q^e \right) \cdot \left( \sum_{q \in Q} \alpha_q^{j+1} \cdot x_q^e \right) \cdot \left[ \sum_{q \in Q} \alpha_q^{j+1} \cdot x_q^e \cdot \omega_q - \sum_{q \in Q} \alpha_q^j \cdot x_q^e \cdot \nu_q - 11 \right] \geq 0 \quad (10)$$

$$\forall e \in E, \forall j \in Start_{week}, \sum_{j'=j}^{j+6} \sum_{q \in Q} \alpha_q^{j'} \cdot x_q^e \cdot l_q \leq 48 \quad (11)$$

The constraints (2), (3) and (4) ensure that employee  $e$  is assigned to shift  $q$  if and only if  $e$  is available on the corresponding day, has the appropriate skill and can work on the corresponding area. Equations (5) and (6) ensure that an employee does not work more than one shift a day and that every shift is assigned to a single employee. The constraints (7) and (8) ensure that the assignment respects the imperative start and end times for each employee. Constraints (9), (10) and (11) relate to the laws that prohibit for each employee, more than 6 consecutive working days (9), less than 11 hours between two shifts (10) and more than 48 hours of work for a week (11). The objective function takes into account the following costs:

- **Under coverage:**  $\$UnderC$ . The cost of under coverage is established for non-assigned shifts. These costs are computed according to the shift duration.

$$\$UnderC = \sum_{q \in Q} (1 - \sum_{e \in E} x_q^e) \cdot \$q \quad (12)$$

- **Over time scheduling:**  $\$OverT$ . The cost of over time scheduling is expressed by the sum of costs related to scheduled working hours above the contractual working hours  $MH_e$  for every employee  $e$ .

$$\$OverT = \sum_{e \in E} \max(0, (\sum_{q \in Q} x_q^e \cdot l_q - MH_e)) \cdot \$sup_e \quad (13)$$

- **Under time scheduling:**  $\$UnderT$ . The cost of under time scheduling is expressed by the sum of cost related to remaining working hours to reach the contractual working hours.

$$\$UnderT = \sum_{e \in E} \max(0, (MH_e - \sum_{q \in Q} x_q^e \cdot l_q)) \cdot \$inf_e \quad (14)$$

- **Employee dissatisfaction:**  $\$EmpD$ . Some social criteria affect the evaluation of the solution, such as allowing employees to be free on weekends, avoiding isolated days-on or days-off. An *isolated day-on* is a working day followed and preceded by at least one day-off. An *isolated day-off* is an off day followed and preceded by at least one working day. This cost cannot be more detailed for confidential reasons.

### 2.3. Solution and operator associated

A solution for our problem is defined as the assignment of shifts to employees on the horizon. This solution is defined by a set of couples (shift, employee) but also by non-assigned shifts.

DEFINITION 2.1 *Formally a solution  $S$  is defined as follows:  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  where  $\mathcal{A} \subset Q \times E$ ,  $Q$  is the set of shifts,  $E$  is the set of employees and  $Q_{AV}$  is the set of non-assigned shifts:  $Q_{AV} = \{q \in Q / \forall e \in E, (q, e) \notin \mathcal{A}\}$ .*

Obviously, all solutions considered in this work have to respect work laws and conditions of effective organization described by the constraints in section 2.2.

DEFINITION 2.2 *A solution  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  is said **feasible** if the constraints (2) to (11) in section 2.2 are respected.*

In order to build a solution, some operators are needed:

DEFINITION 2.3  **$\oplus$  operator.** *Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  be a feasible solution and  $(q, e) \in Q_{AV} \times E$  then  $S \oplus (q, e) = \langle \mathcal{A} \cup \{(q, e)\}, Q_{AV} \setminus \{q\}, E \rangle$ .*

This operator has some interesting properties which we shall refer later in the presentation of our method.

**PROPERTY 2.4** Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  be a solution and  $e_0$  a fictive employee ( $e_0 \notin E$ ) then  $\forall q \in Q$ ,  $(q, e_0)$  is an identity element.  $\forall q \in Q$ ,  $S \oplus (q, e_0) = S$ .

**PROPERTY 2.5** Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  be a feasible solution,  $(q_1, e_1) \in Q_{AV} \times E$ ,  $(q_2, e_2) \in Q_{AV} \times E$  with  $q_1 \neq q_2$ . If  $S \oplus (q_1, e_1)$  and  $S \oplus (q_2, e_2)$  are feasible solutions then:

- if  $e_1 \neq e_2$ ,  $(S \oplus (q_1, e_1)) \oplus (q_2, e_2)$  is feasible too.
- else,  $(S \oplus (q_1, e_1)) \oplus (q_2, e_1)$  is feasible iff  $q_1$  and  $q_2$  do not overlap in time and the constraints (9),(10) and (11) in section 2.2 are not violated for  $e_1$ .

**PROPERTY 2.6**  $(S \oplus (q_1, e_1)) \oplus (q_2, e_2) = (S \oplus (q_2, e_2)) \oplus (q_1, e_1)$ .

Indeed by definition  $(S \oplus (q_1, e_1)) \oplus (q_2, e_2) = \langle \mathcal{A} \cup \{(q_1, e_1)\} \cup \{(q_2, e_2)\}, Q_{AV} \setminus \{q_1\} \setminus \{q_2\}, E \rangle$  which is equal to  $\langle \mathcal{A} \cup \{(q_2, e_2)\} \cup \{(q_1, e_1)\}, Q_{AV} \setminus \{q_2\} \setminus \{q_1\}, E \rangle = (S \oplus (q_2, e_2)) \oplus (q_1, e_1)$ .

**DEFINITION 2.7**  $\oplus$  **operator.** Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  be a feasible solution and  $\mathcal{L} = \{(q_1, e_1), \dots, (q_n, e_n)\}$  where  $\forall i \in [1, n]$ ,  $(q_i, e_i) \in Q_{AV} \times E$  and  $q_1 \neq \dots \neq q_n$ . We denote  $S \oplus \mathcal{L} = (\dots (S \oplus (q_1, e_1)) \oplus \dots) \oplus (q_n, e_n)$ .

**PROPERTY 2.8** If  $S \oplus (q_1, e_1), \dots, S \oplus (q_n, e_n)$  are feasible solutions then:

- if  $e_1 \neq \dots \neq e_n$ , by extension of property 2.5  $S \oplus \mathcal{L}$  is feasible too.
- else,  $S \oplus \mathcal{L}$  is feasible iff all shifts  $q_i \in Q_e$  where  $Q_e = \{q_i / (q_i, e) \in \mathcal{L}\}$ , do not overlap in time and the constraints (9), (10) and (11) in section 2.2 are not violated for  $e$ .

**DEFINITION 2.9**  $\ominus$  **operator.** Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  and  $(q, e) \in \mathcal{A}$  then  $S \ominus (q, e) = \langle \mathcal{A} \setminus \{(q, e)\}, Q_{AV} \cup \{q\}, E \rangle$ .

**DEFINITION 2.10**  $\ominus$  **operator.** Let  $S = \langle \mathcal{A}, Q_{AV}, E \rangle$  be a a feasible solution and  $\mathcal{L} = \{(q_1, e_1), \dots, (q_n, e_n)\}$  where  $\forall i \in [1, n]$ ,  $(q_i, e_i) \in Q_{AV} \times E$  and  $q_1 \neq \dots \neq q_n$ . We denote  $S \ominus \mathcal{L} = (\dots (S \ominus (q_1, e_1)) \ominus \dots) \ominus (q_n, e_n)$ .

Properties 2.5, 2.6 and 2.8 can be extended to operators  $\ominus$  and  $\ominus$ .

## 2.4. Quality of a solution

There are many solutions, but not all of them have the same implication from a quality point of view. The quality of solution  $S$  is defined by the objective function  $f(S)$  (see equation 15).

$$f(S) = w_1.\$UnderC(S) + w_2.\$OverT(S) + w_3.\$UnderT(S) + w_4.\$EmpD(S) \quad (15)$$

Terms  $\$UnderC(S)$ ,  $\$OverT(S)$ ,  $\$UnderT(S)$  and  $\$EmpD(S)$  refer to costs defined in equation (1) applied to solution  $S$ . Weights  $w_k$  are used to promote the quantification of

terms and are established according to planning operator expertise. Solving our problem corresponds to finding solution  $S$  such as  $f(S)$  is minimal. Costs of criteria are on the same scale and expressed in homogeneous unit (HU).

### 3. Memetic Algorithm

As a first step, we developed greedy algorithms and local search method to solve the planning problem presented in (Abdoul Soukour *et al.* (2012)). To improve these first results, we have worked out a Memetic Algorithm. Such kind of metaheuristic is recent (Moscato (1999)). It combines Genetic Algorithm and Local Search techniques. The specific concepts of Genetic Algorithms are introduced in (Holland (1992)). A population of solutions (individuals) evolves from a generation to another. The individuals are encoded into a similar structure called a *chromosome*. The individuals of the current generation are combined by a *crossover* operator to produce the individuals of the next one. A diversification process is also used to avoid homogeneity in the population. This diversification is obtained through a *mutation* operation. Local Search steps in this process regarding Memetic Algorithm, not only to diversify but also to improve the current population. To insert an individual into the next population and to identify improvements, it is necessary to know the performance of each individual, its *fitness*, in the population through an evaluation procedure. In our study, this evaluation involves the objective function  $f$  in equation 15 presented in the section 2.4.

#### 3.1. Chromosome encoding

To encode solutions, we consider a fixed order of shifts in  $Q$ , denoted  $Q_0 = \langle q_1, q_2, \dots, q_{|Q|} \rangle$ . Let  $S_k = \langle A, Q_{AV}, E \rangle$  be a feasible solution, then  $S_k$  is encoded by chromosome  $C_k = \langle (q_1, e_1), (q_2, e_2), \dots, (q_{|Q|}, e_{|Q|}) \rangle$ , respecting  $Q_0$  order, such that  $e_j$  is an employee of  $E$  if the assignment  $(q_j, e_j) \in A$  and  $e_j = e_\emptyset$ , the fictive employee, otherwise ( $q_j \in Q_{AV}$  in that case). Obviously,  $e_i$  and  $e_j$  with  $i \neq j$  could be the same employee, because one employee works more than once on the horizon. We notice that for any two different chromosomes the sequences of the shifts are always the same. So, we can reduce the expression of  $C_k$  to the sequence of employees  $\langle e_1, e_2, \dots, e_{|Q|} \rangle$ . We will denote  $C_k[i] = e_i$  the  $i$ -th *gene* of the chromosome  $C_k$ . At each solution  $S_k$  corresponds a chromosome  $C_k$  and vice versa, and the fitness of  $C_k$  is  $f(S_k)$ . We notice that our encoding is a *direct* one, because chromosomes fully describe solutions.

EXAMPLE 3.1 *As an illustration, the solution*

$$S_1 = \langle \{(q_1, e_b), (q_3, e_a), (q_4, e_b)\}, \{q_2\}, \{e_a, e_b\} \rangle$$

*is represented by the chromosome  $C_1 = \langle e_b, e_\emptyset, e_a, e_b \rangle$ .*

#### 3.2. Selection Operator

In order to provide a new generation, we need to recombine some chromosomes. In our method we chose to elect two chromosomes to create a new one. There are many ways to chose the parents (roulette-wheel selection, truncation selection, tournament selection, etc.). In this work, we use the tournament selection (Prins (2004)).  $k$  individuals are

randomly selected and the one with the best fitness is retained as first parent, chromosome  $C_1$ . Ditto for the second parent, chromosome  $C_2$ . Objective is to transmit good properties of parents to offspring. These parents must be different.

### 3.3. Crossover Operator

In order to produce new solutions we use Crossover Operator by first selecting two parents among the population and constructing one new individuals by combining their genes. Parents are chosen as described previously. There exist numerous types of crossover operator in the literature, depending on the considered problems like the Traveling Saleman Problem (Oliver *et al.* (1987)) or the Scheduling problems (Portmann and Vignier (2000)) and obviously depending on their encoding. We used an operator adapted to our problem that is a parallel version of the well known 1-point crossover operator. The main idea is to compare the  $i$ -th gene of the first parent with the  $i$ -th gene of the second parent and transmit the best one to the offspring, starting from a random position in the chromosomes.

Let  $C_1$  and  $C_2$  be the parent chromosomes and  $C_{neo}$  be their child under construction, we note  $C_{neo} = \text{Crossover}(C_1, C_2)$ . Initially  $C_{neo} = \langle e_\emptyset, e_\emptyset, \dots, e_\emptyset \rangle$  and the associated solution is  $S_{neo} = \langle \emptyset, Q, A \rangle$ . Let  $X$  be a random value in  $[1, |Q|]$ . The genes of the same rank of  $C_1$  and  $C_2$  are compared from the  $X$ -th rank in a circular manner. The best of both genes, subject to feasibility, is transmitted to  $C_{neo}$ . Algorithm 1 gives the details of the construction of  $C_{neo}$  by our crossover operator. The complexity of this algorithm is  $O(|Q|)$ .

**EXAMPLE 3.2** For example, let  $Q = \{q_1, q_2, q_3\}$  be the set of available shifts,  $E = \{e_b, e_b\}$  be the set of available employees,  $C_1 = \langle e_a, e_\emptyset, e_a \rangle$  and  $C_2 = \langle e_b, e_a, e_\emptyset \rangle$  be the parent chromosomes. Suppose that  $X = 2$ .

Initially  $C_{neo} = \langle e_\emptyset, e_\emptyset, e_\emptyset \rangle$  and  $S_{neo} = \langle \emptyset, \{q_1, q_2, q_3\}, \{e_a, e_b\} \rangle$ .

- (1) First we compare  $f(S_{neo} \oplus (q_2, C_1[2]))$  and  $f(S_{neo} \oplus (q_2, C_2[2]))$ , suppose that both are feasible and  $f(S_{neo} \oplus (q_2, C_1[2])) > f(S_{neo} \oplus (q_2, C_2[2]))$ , then  $C_{neo} = \langle e_\emptyset, e_a, e_\emptyset \rangle$  and  $S_{neo} = \langle \{(q_2, e_a)\}, \{q_1, q_3\}, E \rangle$
- (2) Next we compare  $f(S_{neo} \oplus (q_3, C_1[3]))$  and  $f(S_{neo} \oplus (q_3, C_2[3]))$ , suppose that only  $S_{neo} \oplus (q_3, C_1[3])$  is feasible, then  $C_{neo} = \langle e_\emptyset, e_a, e_a \rangle$  and  $S_{neo} = \langle \{(q_2, e_a), (q_3, e_a)\}, \{q_1\}, E \rangle$
- (3) At least we compare  $f(S_{neo} \oplus (q_1, C_1[1]))$  and  $f(S_{neo} \oplus (q_1, C_2[1]))$ , suppose that both  $S_{neo} \oplus (q_1, C_2[1])$  and  $S_{neo} \oplus (q_1, C_1[1])$  are not feasible then  $C_{neo} = \langle e_\emptyset, e_a, e_a \rangle$  and  $S_{neo} = \langle \{(q_2, e_a), (q_3, e_a)\}, \{q_1\}, E \rangle$

So the new chromosome is  $C_{neo} = \langle e_\emptyset, e_a, e_a \rangle$ .

### 3.4. Initialization of the Population

Memetic Algorithm has to start with a population of consistent chromosomes, i.e., each chromosome must be a feasible solution (see property 2.2). To create a diversified population, we used three algorithms to provide propitious individuals and a random generation for the rest of the population.

First algorithms are a greedy algorithm Gshifts and a daily global assignment algorithm AFshift (Abdoul Soukour *et al.* (2012)). These algorithms consist in constructing one solution by adding, step by step, an assignment  $(q, e)$  for Gshifts or a subset of

---

**Algorithm 1** Crossover

---

**Input:** $Q$ : set of available shifts $E$ : set of available employees $C_1$ : chromosome $C_2$ : chromosome $X$ : random position in chromosome  $[1, |Q|]$ **Output:** $C_{neo}$ : chromosome $C_{neo} \leftarrow \langle e_\emptyset, e_\emptyset, \dots, e_\emptyset \rangle$  $S_{neo} \leftarrow \langle \emptyset, Q, E \rangle$  the solution associated to  $C_{neo}$ **for**  $r = 0, \dots, |Q| - 1$  **do** $i \leftarrow ((r + (X - 1)) \text{ modulo } |Q|) + 1$ **if**  $S_{neo} \oplus (q_i, C_1[i])$  is feasible **then****if**  $S_{neo} \oplus (q_i, C_2[i])$  is feasible **then****if**  $f(S_{neo} \oplus (q_i, C_1[i])) \leq f(S_{neo} \oplus (q_i, C_2[i]))$  **then** $C_{neo}[i] \leftarrow C_1[i]$  $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_1[i])$ **else** $C_{neo}[i] \leftarrow C_2[i]$  $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_2[i])$ **end if****else** $C_{neo}[i] \leftarrow C_1[i]$  $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_1[i])$ **end if****else****if**  $S_{neo} \oplus (q_i, C_2[i])$  is feasible **then** $C_{neo}[i] \leftarrow C_2[i]$  $S_{neo} \leftarrow S_{neo} \oplus (q_i, C_2[i])$ **end if****end if****end for****return**  $C_{neo}$ 

---

assignments related to a day for AFshifts. We also use an improvement solution algorithm IDCshifts (on solutions provided by AFshift or Gshift) to provide chromosomes with good quality. It is based on an iterative Destruction/Construction algorithm which combines destruction and construction steps to explore the space of the solution neighbourhood.

**Gshifts.** The principle of *Gshifts* is to construct a path of acceptable shifts for each employee on the horizon. For each employee  $e \in E$  and for each day  $j$  of the horizon, we consider the set of non-assigned shifts scheduled for day  $j$ . Among these shifts,  $q$  is chosen such that  $S \oplus (q, e)$  is feasible and  $f(S \oplus (q, e))$  is minimal. As the arrangement of employees affects the structure of the solution, they are randomly considered. Insertion cost evaluation is done in  $O(1)$ . The complexity of Gshifts is  $O(|E| \times |Q|)$ .

---

**Algorithm 2** IDCshifts

---

**Input:** $S_0$ : initial solution $D_{max}$ : number of assignments to destroy**Output:** $S_{best}$ : best solution $S \leftarrow S_0$  $S_{best} \leftarrow S$ **repeat** $d \leftarrow$  random choice between 1 and  $D_{max}$  $S_{dest}^d \leftarrow Destruction(S, d)$ **if**  $f(S_{dest}^d) < f(S)$  **then** $S_{best} \leftarrow S_{dest}^d$ **end if** $S_{const} \leftarrow Construction(S_{dest}^d)$ **if**  $f(S_{const}) < f(S_{dest}^d)$  **then** $S_{best} \leftarrow S_{const}$ **end if****until** runtime not expired**return**  $S_{best}$ 

---

**AFshifts.** It computes an optimal assignment (with minimal cost) for each day  $j$ , between a set of shifts scheduled for day  $j$ ,  $Q_{AVj} \subseteq Q$  and a set of available employees for this day  $j$ ,  $E_{AVj} \subseteq E$ . So the global problem is decomposed day by day. Also in this case, the day order examination influences the structure of the solution, and so, they are randomly considered.

Let  $S_k$  be the current solution constructed after  $k$  iterations and consider we will construct the planning of day  $j$ . For each couple  $(q, e)$ ,  $q \in Q_{AVj}$ ,  $e \in E_{AVj}$ , assignment cost  $f(S_k \oplus (q, e))$  is computed in  $O(1)$ .  $f(S_k \oplus (q, e)) = \infty$  if  $S_k \oplus (q, e)$  is not a feasible solution. A classical resolution of assignment problem such as Hungarian algorithm (Kuhn (1955)) allows to compute a maximum matching, with minimal cost, between shifts and employees for the considered day. It provides a subset of feasible assignments  $\mathcal{L} = \{(q_1, e_1), \dots, (q_x, e_x)\}$ . Because these assignments are feasible  $S_k \oplus \mathcal{L}$  is a feasible solution (see property 2.8). AFshifts gives a solution in  $O(|J| \times |E|^3)$ .

**IDCshifts.** The algorithm (see algorithm 2) is based on the principle of the iterative Destruction/Construction algorithm proposed in (Ruiz and Stützle (2007), Bouly *et al.* (2008), Abdoul Soukour *et al.* (2012)). As it is an improvement solution algorithm, it works on an existing solution  $S = \langle A, Q_{AV}, E \rangle$ . From  $S$ , it consists in first removing a subset of  $d$  assignments from  $S$ . These  $d$  removed assignments, are randomly chosen.  $d$  is also randomly chosen between 1 and  $\frac{|Q|}{|E|}$ . For each destruction of assignment  $(q, e)$ , the solution  $S \ominus (q, e)$  respects all constraints (7) to (11) of section 2.2. Let  $\mathcal{L}$  be the set of the  $d$  assignments to remove from  $S$ . We denote  $S_{dest}^d = S \ominus \mathcal{L}$ .

In a second step, the algorithm rebuilds solution, starting from  $S_{dest}^d$  by *Construction* process, that is an adaptation of the best insertion method (Ruiz and Stützle (2007)). This step consists in computing new assignments from available shifts (including available shifts provided by destruction step) and available employees (including those provided



by destruction step). For each day  $j$  on the horizon, let  $Q_j$  be the set of non assigned shifts of day  $j$  and  $E_j$  the set of available employees of day  $j$ . For each day  $j$  and for each shift  $q \in Q_j$ , we determine the available employee  $e \in E_j$  such as adding  $(q, e)$  provides best improvement of the solution quality.  $(q, e)$  is added to  $S_{dest}^d$ . Note that days, shifts and employees are randomly examined.

These two steps are alternatively iterated until a preset runtime is expired. The complexity of this algorithm is  $O(|Q| \times |E|)$ . *IDCshifts* will be used in our experimentations to evaluate the improvement of our Memetic Algorithm (i.e.  $f(S_{IDCshifts})$  will be the reference value).

### 3.5. Local Search as Improvement Operator

In GA, when a new chromosome is generated, mutation operator is used to alter some genes with a probability equal to mutation rate  $pm$ . The aim is to introduce some extra variability into the population. For the *MA*, this mutation is processed by Local Search (LS) techniques whose purpose can be the improvement of the considered solution. In our algorithm this Improvement Operator is a combination of three neighborhood functions dedicated to exploring the solution space differently. They are denoted *REAS*, *COMP* and *DC*. They are applied, alternatively and randomly, until there is no further improvement, as described in algorithm 3. The shared principle of these functions is to compute new feasible assignments of shifts to employees, with or without prior destructions. The resulting solution is always better than or equivalent to the original one.

---

#### Algorithm 3 Improvement Operator

---

**Input:**

$C_0$ : initial chromosome { its associated solution  $S_0$  }

**Output:**

$C$ : improved chromosome { its associated solution  $S$  }

$S \leftarrow S_0$

**repeat**

$ImpOP \leftarrow$  random choice between unmarked functions in  $\{REAS, COMP, DC\}$

$S' \leftarrow ImpOP(S)$

Mark the function corresponding to  $ImpOP$

**if**  $f(S') < f(S)$  **then**

$S \leftarrow S'$

Unmark all functions

**end if**

**until** There is no unmarked function

**return**  $C$  {the associated chromosome of  $S$ }

---

**Reassignment operator REAS.** It consists in randomly extracting from each employee  $e$ , an assignment  $(q, e) \in A$  from the selected solution  $S = \langle A, Q_{AV}, E \rangle$ . We denote  $S' = S \ominus (q, e)$ . Then, for each of these extractions, *REAS* rebuilds a solution  $S' \oplus (q', e)$  such that  $q' \in Q_{AV} \cup \{q\}$ ,  $S' \oplus (q', e)$  is feasible and  $f(S' \oplus (q', e))$  is minimal. We note that, only employees with at least one assigned shift, are considered. The returned solution  $S_{REA}$ , has a potential different set of assignments, but an equal number

of non-assigned shifts and  $f(S_{REA}) \leq f(S)$  (see example 3.3). The complexity of this process is  $O(|Q| \times |E|)$ .

**EXAMPLE 3.3** Let  $S = \langle \{(q_1, e_b), (q_3, e_a), (q_4, e_b)\}, \{q_2\}, \{e_a, e_b, e_c\} \rangle$  be the new individual. We consider  $S'_{e_a} = S \ominus (q_3, e_a)$  and  $S'_{e_b} = S \ominus (q_4, e_b)$  where  $(q_3, e_a)$  (respectively  $(q_4, e_b)$ ) has been randomly chosen. Moreover  $S'_{e_a} \oplus (q_2, e_a)$  and  $S'_{e_b} \oplus (q_2, e_b)$  are feasible. Then the operator REAS explore the following neighbor solutions:

- $\langle \{(q_1, e_b), (q_2, e_a), (q_4, e_b)\}, \{q_3\}, \{e_a, e_b, e_c\} \rangle$
- $\langle \{(q_1, e_b), (q_2, e_a), (q_3, e_b)\}, \{q_4\}, \{e_a, e_b, e_c\} \rangle$

Note that  $e_c$  is not considered since there is no shift associated to it.

**Complete operator COMP.** It consists in attempting to add a new improving assignment  $(q', e)$ , for each employee  $e$  such that there exists an assignment  $(q, e) \in A$  of the selected solution  $S = \langle A, Q_{AV}, E \rangle$  with  $q' \in Q_{AV}$  and  $S \oplus (q', e)$  is feasible. The returned solution  $S_{COMP}$ , has a potential increased set of assignments, and a smaller or equal number of non-assigned shifts and  $f(S_{COMP}) \leq f(S)$  (see example 3.4). The complexity of this process is  $O(|Q| \times |E|)$ .

**EXAMPLE 3.4** Let  $S = \langle \{(q_1, e_b), (q_3, e_a), (q_4, e_b)\}, \{q_2\}, \{e_a, e_b, e_c\} \rangle$  be the new individual. We consider that  $(q_2, e_a)$  and  $(q_2, e_b)$  are feasible. Then the operator COMP explore the following neighbor solutions:

- $\langle \{(q_1, e_b), (q_3, e_a), (q_4, e_b), (q_2, e_a)\}, \emptyset, \{e_a, e_b, e_c\} \rangle$
- $\langle \{(q_1, e_b), (q_2, e_a), (q_4, e_b), (q_2, e_b)\}, \emptyset, \{e_a, e_b, e_c\} \rangle$

Note that employee  $e_c$  is not considered too with this operator.

**Destruction Construction operator DC.** The Destruction Construction operator is based on the principle of the Iterative Destruction Construction algorithm presented previously in algorithm 2. It consists in doing one iteration of *IDC shifts*. The complexity of this process is  $O(|Q| \times |E|)$ .

**EXAMPLE 3.5** Let  $S = \langle \{(q_1, e_b), (q_2, e_a), (q_3, e_b), (q_4, e_c), (q_6, e_a)\}, \{q_5, q_7, q_8\}, \{e_a, e_b, e_c\} \rangle$  be the new individual. The number of removed assignment is  $d = 2$ . Then the two assignments  $(q_2, e_a)$  and  $(q_4, e_c)$  are randomly chosen and removed. We obtain the new solution  $S_0 = \langle \{(q_1, e_b), (q_3, e_b), (q_6, e_a)\}, \{q_2, q_4, q_5, q_7, q_8\}, \{e_a, e_b, e_c\} \rangle$ . We suppose that the horizon is days  $j1$ ,  $j2$  and  $j3$ . Then the construction process explores the solution space as follows:

- Day  $j3$ :  $Q_{j3} = \{q_7, q_8\}$  and  $E_{j3} = \{e_a, e_b\}$ .
  - DC compares  $f(S \oplus (q_7, e_a))$  and  $f(S \oplus (q_7, e_b))$  (both are feasible solutions), then retains  $S \oplus (q_7, e_b)$
  - DC retains  $S \oplus (q_8, e_a)$  because  $e_b$  is not yet available for day  $j3$ .
- Day  $j1$ :  $Q_{j1} = \emptyset$  and  $E_{j1} = \{e_a, e_c\}$ .
- Day  $j2$ :  $Q_{j2} = \{q_2, q_4, q_5\}$  and  $E_{j2} = \{e_a, e_c\}$ .
  - DC compares  $f(S \oplus (q_5, e_a))$  and  $f(S \oplus (q_5, e_c))$  (both are feasible solutions), then retains  $S \oplus (q_5, e_a)$
  - DC retains  $S \oplus (q_2, e_c)$  because  $e_a$  is not yet available for day  $j2$ .
  - No more available employee for shift  $q_4$

We remind that days, shifts and employees are randomly examined. The resulting solution is:  $S = \langle \{(q_1, e_b), (q_2, e_c), (q_3, e_b), (q_5, e_a), (q_6, e_a), (q_7, e_b), (q_8, e_a)\}, \{q_4\}, \{e_a, e_b, e_c\} \rangle$

---

**Algorithm 4** Memetic Algorithm (*MA*)

---

**Input:**

$Q$ : set of shifts  
 $E$ : set of employees  
 $N$ : population size  
 $pm_0$ : initial probability  
 $\lambda$ : constant

**Output:**

$S_{best}$ : best solution found

$P \leftarrow Initialization(N) \{P = \{C_0, C_1, \dots, C_{N-1}\}\}$

$P \leftarrow DescendingSort(P, f)$

$pm \leftarrow pm_0$

**while** *runtime not expired* **do**

$C' \leftarrow Selection(P)$

$C'' \leftarrow Selection(P \setminus \{C'\})$

$C_{neo} \leftarrow Crossover(C', C'')$

$prob \leftarrow$  random between 0 and 1

**if**  $prob \leq pm$  **then**

$C_{neo} \leftarrow Improvement(C_{neo})$

**end if**

**if**  $f(S_{neo}) \leq f(S_0)$  **then**

**if**  $\exists k \in [0, N-1] | f(S_{neo}) = f(S_k)$  **then**

$C_k \leftarrow C_{neo}$

$pm \leftarrow \lambda \times pm$

**else**

            Eject  $C_0$  from  $P$

            Insert  $C_{neo}$  in  $P$

$pm \leftarrow pm_0$

**end if**

**else**

$pm \leftarrow \lambda \times pm$

**end if**

**end while**

$S_{best} \leftarrow S_{N-1}$

**return**  $S_{best}$

---

### 3.6. General Algorithm

The Memetic Algorithm is described by Algorithm 4. First, population  $P$  is initialized as mentioned in section 3.4 with  $N$  chromosomes  $C_0, C_1, \dots, C_{N-1}$ . We note  $f(S_k)$  the *fitness* associated to the chromosome  $C_k$ . Generally,  $C_k$  always designates the chromosome associated with  $S_k$  and vice versa. A change in one causes the modification of the other implicitly. For more convenience, the population is sorted by descending fitness values  $f$  such that  $\forall i \in [0, N-2], f(S_i) \geq f(S_{i+1})$ .

At each iteration of the algorithm a couple of individuals,  $C'$  and  $C''$ , is chosen among the population using *Selection* operator (see section 3.2). Then *Crossover* operator (see section 3.3) is used to produce a child chromosome  $C_{neo}$  from them.  $C_{neo}$  could be improved by *Improvement* operator (see section 3) with probability  $pm$  (initially preset to  $pm_0$ ).

When  $C_{neo}$  ameliorates the population, then it integrates population. There is two

$Num$	$E$	$C$	$Q$	$QMax$	$ContH$	$\frac{Q}{E}$	$\bar{Q}_{day}$
1	277	6	2343	141	14	8.46	75.58
2	197	6	1998	95	2	10.14	64.45
3	297	6	2425	153	14	8.16	78.23
4	224	5	2206	109	1	9.85	71.16
5	323	5	2065	109	1	6.39	66.61
6	289	3	1625	92	3	5.62	42.42
7	148	5	1089	57	2	7.36	35.13
8	32	3	496	16	2	15.50	16.00
9	294	6	3449	197	1	11.73	111.26
10	104	5	1027	55	2	9.88	33.13
11	48	5	713	23	3	14.85	23.00
12	241	6	3777	122	14	15.67	121.84
13	227	6	3818	125	6	16.82	123.16

Table 1. Instances characteristics

possible cases for this configuration: either there is a chromosome  $C_k$  with a same fitness, then  $C_{neo}$  replaces  $C_k$ , or  $C_{neo}$  strictly improves the population, then the worst chromosome  $C_0$  is removed and  $C_{neo}$  is inserted.

Probability of child improvement,  $pm$ , evolves following the *fitness* of the new generated individuals. While there is no strict improvement  $pm$  decreases ( $pm = \lambda \times pm$ ,  $0 < \lambda < 1$ ), but once the population is improved  $pm$  is reset to  $pm_0$ . To sum up, population P is changed step by step, and the process is iterated for a fixed running time. Algorithm *MA* returns solution  $S_{best}$  corresponding to the best chromosome of the population,  $C_{N-1}$ , at the end of the process.

#### 4. Numerical results

All algorithms have been implemented in java and tested on Intel Xeon Quad Core at 2.4 Ghz. They have been trained on instances which are extracted from real cases.

**Instances.** The main characteristics of our 13 instances are given in Table 1. Columns are respectively,  $Num$ , instance number,  $E$ , number of employees,  $C$  number of skills,  $Q$ , number of shifts,  $QMax$ , maximum number of shifts for a day,  $ContH$ , number of kinds of employment contracts,  $\frac{|Q|}{|E|}$ , average number of shifts per employee and  $\bar{Q}_{day}$ , average number of shifts per day. The rostering horizon considered for instances is a month (from 28 to 31 days). Instances cannot be more detailed due to confidential reasons.

**MA experimentations.** To set the most appropriate population size for this problem, we tested *MA* for different values, and compare the improvement gap from a basic solution supplied by *IDCshifts* and denoted  $S_{IDCshifts}$  (see section 3.4). We denote  $S_{MA(x)}$  the best solution exhibited by *MA* when the population size is set to  $x$ . Percentage of improvement obtained by *MA* is expressed by equation 16. Results for sizes 5, 10, 15, 20 and 25 are reported in table 2. Populations are initialized as described in section 3.4. *MA* was performed 10 times during 30 minutes for each instance. Sub-column *max* is related to the best improvement among all executions, and *avg* is related to the average

Num	MA(5) $\varepsilon_{IDCshifts}^{MA(5)}$ (%)		MA(10) $\varepsilon_{IDCshifts}^{MA(10)}$ (%)		MA(15) $\varepsilon_{IDCshifts}^{MA(15)}$ (%)		MA(20) $\varepsilon_{IDCshifts}^{MA(20)}$ (%)		MA(25) $\varepsilon_{IDCshifts}^{MA(25)}$ (%)	
	max	avg	max	avg	max	avg	max	avg	max	avg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.87	0.73	0.85	0.63	0.97	<b>0.76</b>	0.78	0.62	<b>0.96</b>	0.75
8	51.65	49.76	49.55	47.63	50.59	48.64	<b>51.94</b>	<b>50.26</b>	51.81	49.40
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	<b>4.14</b>	<b>3.55</b>	3.33	2.85	3.30	2.98	3.94	3.00	3.18	2.50
11	32.08	30.91	33.95	32.60	35.45	32.30	<b>35.69</b>	<b>33.30</b>	34.16	32.40
12	<b>52.76</b>	49.52	51.92	50.06	50.46	48.84	51.58	<b>50.26</b>	50.29	48.84
13	43.31	<b>41.46</b>	42.60	40.94	42.52	41.05	<b>43.99</b>	41.09	41.48	39.20

Table 2. *MA* best improvement and improvement on average compared with *IDCshifts* solution w.r.t. population size

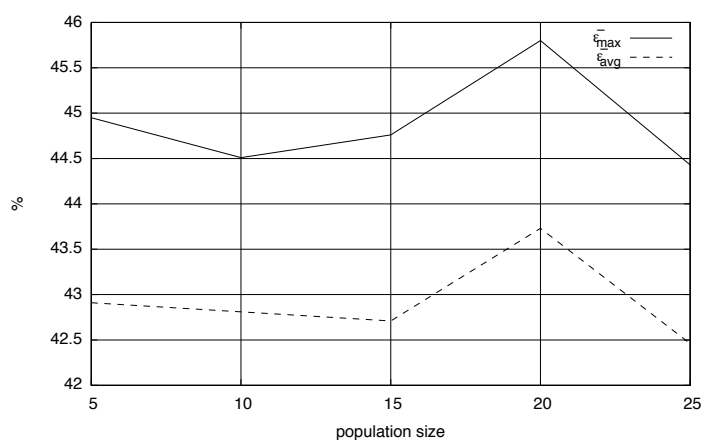


Figure 3. Improvement over the size of the population

improvement. The mutation rate is  $pm_0 = 0.75$  and its decreasing ratio is  $\lambda = 0.99$ . The  $D_{max}$  value of operator DC is fixed to  $\frac{|Q|}{|E|}$ .

$$\varepsilon_{IDCshifts}^{MA(x)} = \frac{f(S_{IDCshifts}) - f(S_{MA(x)})}{f(S_{IDCshifts})} \quad (16)$$

It clearly appears that *MA* did not improve the results for the instances 1 to 6 & 9. Indeed, *IDCshifts* provides good results for these instances, that can be explained by the large difference between  $Q_{day}$  and  $Q_{Max}$ . The greater is this difference, the less the workload is uniform on the horizon. Indeed, the main *IDCshifts* feature is a random exploration of solution space. Moreover as  $\frac{|Q|}{|E|}$  is small for these instances the assignment optimization is easier.

Nevertheless the improvement is significant for instances 8, 11, 12 and 13 (more than 50% for instance 12). The gap between  $Q_{day}$  and  $Q_{Max}$  (see table 1) is closed to zero that reveals a uniform workload structure. So for a given available shift there are more

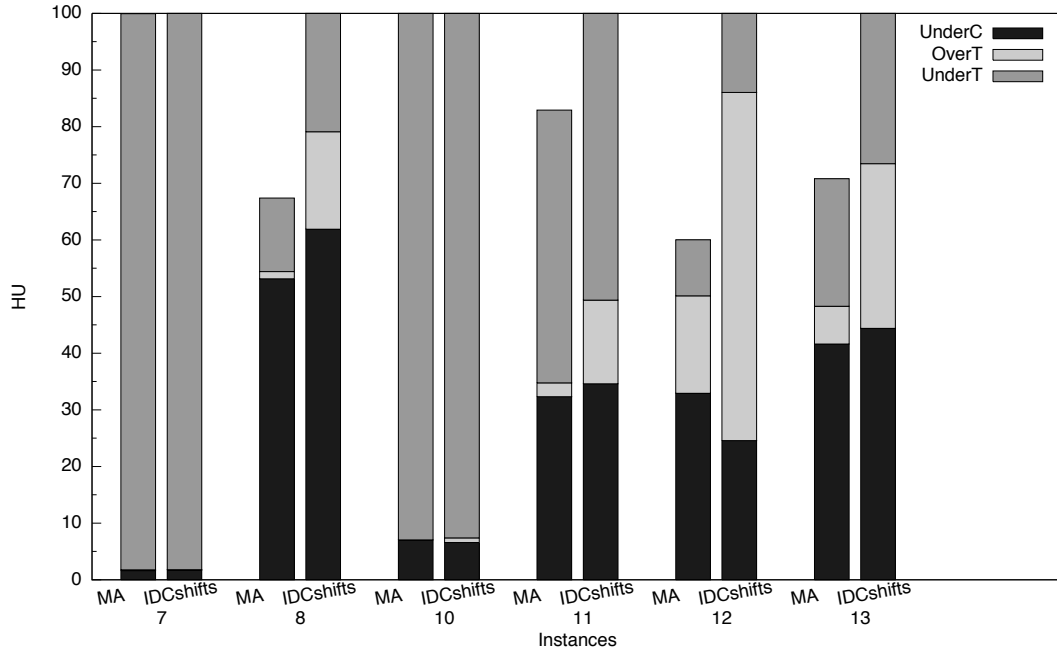


Figure 4. Distribution of cost criteria

eligible employees that allows to explore a larger neighborhood for *REAS* and *COMP* functions. Moreover both are descent methods then improve the solution.

For instances 7 & 10, the improvement is weak as the gap between  $\bar{Q}_{day}$  and  $QMax$  is closed to easier instances. In the following, only significant instances (7, 8, 10, 11, 12 and 13) will be considered.

In figure 3,  $\bar{\epsilon}_{\max(x)}$  (respectively  $\bar{\epsilon}_{\text{avg}(x)}$ ) represents the mean of instance best improvements (respectively the mean of instance improvements on average) for population of size  $x$ . It brings out that a population of 20 individuals is best suited for instances 8, 11 & 12. For other instances, the gap between the best improvement and improvement obtained for a population size of 20 individuals is tiny. So choose a population size of 20 individuals as reference is convenient for our staff scheduling problem.

**Result Analysis: distribution of cost criteria.** In this part, we studied the impact of three cost criteria:  $\$UnderC$ ,  $\$OverT$  and  $\$UnderT$  (see equation 15) on *MA* solutions. The fourth criterion  $\$EmpD$  cannot be more detailed due to confidential reasons.

Figure 4 presents a comparison between *IDCshifts* and *MA* best solutions. For each instance, we set the cost of *IDCshifts* solution to 100 U.H. and proportionally distribute the criteria costs. Then we have reported the *MA* solution cost relatively to *IDCshifts* one. We can now discuss about the *MA* and *IDCshifts* cost distributions. We notice that *MA* process tends to improve  $\$OverT$  and  $\$UnderT$  criteria. Cost  $\$UnderC$  is slightly improved and even deteriorated for instance 12 that presented a lot of overtime in the *IDCshifts* solution. Distribution of these costs is an interesting indication for the company, both for rewrite and refine the *MA* (change weights  $w_k$  in section 2.4 to guide *MA* to other kinds of solutions), but also as tools for management.

## 5. Conclusion & Perspectives

In this paper, we have proposed a Memetic Algorithm *MA* for solving a realistic scheduling problem, in the area of airport security service, with a lot of specificities. We first have modeled the problem and give a formal definition of the solutions. We used our previous dedicated heuristics *Gshift*, *AFshift* and *IDCshifts* to generate the initial population and a basic tournament selection operator to pick the parents in the reproduction process. We have presented a devoted crossover operator and a suitable improvement operator, made of three neighborhood functions: *REAS*, *COMP* and *DC*.

Our *MA* algorithm proved its efficiency by the quality of provided plannings. In addition we have defined an economic function that allows to evaluate planning with relevance. Relatively to this function, *MA* always provides better solution than those produced by planning operators. Let us note that the planning operators manually adapt solutions supplied by the current software. Perspectives related to this work are various. Among them, other elementary operations on planning could be defined but also others neighborhoods could be included for our Memetic Algorithm.

## Acknowledgements

We thank Mr. Vathana TIV, Senior Director Operational Efficiency of ICTS Europe Holdings BV, for his contribution in this project.

## References

- Abdoul Soukour, A., *et al.*, 2012. Staff Scheduling in Airport Security Service. *In: 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'12)*.
- Aparecido Constantino, A., Marcondes Filho, W., and Landa-Silva, D., 2010. Iterated Heuristic Algorithms for the Classroom Assignment Problem. *In: PATAT'10*, 152–166.
- Ásgeirsson, E.I., 2010. Bridging the gap between self schedules and feasible schedules in staff scheduling. *In: PATAT'10*, 81–96.
- Aykin, T., 1998. A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *JORS*, 49 (6), 603–615.
- Bäumelt, Z., Šúcha, P., and Hanzálek, Z., 2010. An Evolutionary Algorithm in a Multi-stage Approach for an Employee Rostering Problem with a High Diversity of Shifts. *In: PATAT'10*, 97–110.
- Bechtold, S. and Jacobs, L., 1990. Implicit model of flexible break assignments, in optimal shift scheduling. *Management Science*, 36 (11), 1339–1351.
- Blöchliger, I., 2004. Modeling staff scheduling problems. A tutorial. *European Journal of Operational Research*, 158, 533–542 Invited Tutorial.
- Boris, D., *et al.*, 2009. Cut generation for an employee timetabling problem. *European Journal of Operational Research*, 197, 1178–1184.
- Bouly, H., Dang, D.C., and A. Moukrim, A., 2010. A memetic algorithm for the team orienteering problem. *4OR*, 8 (1), 49–70.
- Bouly, H., *et al.*, 2008. Un algorithme de destruction/construction itératif pour un problème de tournées de véhicules spécifique. *In: Conférence Internationale de Modélisation, Optimisation et SIMulation*.

- Ernst, A.T., *et al.*, 2004. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153, 3–27.
- Garey, M.R. and Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
- Holland, J.H., 1992. Adaptation in natural and artificial system. *In: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.*, 219–234 MIT Press.
- Kragelund, L. and Kabel, T., Employee Timetabling. Master’s thesis, Department of Computer Science, University of Aarhus, Denmark An Empirical Study, 1998. .
- Kuhn, H.W., 1955. The Hungarian Method for the assignment problem. *Naval Research Logistic Quarterly*, 2, 83–97.
- Moscato, P., 1999. Memetic Algorithms: a short introduction. *In: New Ideas in Optimization.*, 219–234 McGraw-Hill.
- Métivier, J.P., Boizumault, P., and Loudni, S., 2010. Combining VNDS with Soft Global Constraints Filtering for Solving NRPs. *In: PATAT’10*, 259–272.
- Oliver, I., Smith, D., and Holland, J., 1987. A study of permutation crossover operators on the traveling salesman problem. *In: Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 224 – 230.
- Portmann, M.C. and Vignier, A., 2000. Performances’ study on crossover operators keeping good schemata for some scheduling problems. *In: Genetic and Evolutionary Computation Conference*, 331–338.
- Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & OR*, 31 (12), 1985–2002.
- Rekik, M., Cordeau, J.F., and Soumis, F., 2004. Using benders decomposition to implicitly tour scheduling. *Annals of Operations Research*, 128, 111–133.
- Ruiz, R. and Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177, 2033–2049.
- Soumis, F., Pesant, G., and Rousseau, L.M., 2005. 4: Gestion des horaires et affectation du personnel. *In: Gestion de production et ressources humaines.*, 71–110 Presses Internationales Polytechnique.
- Tien, J. and Kamiyama, A., 1982. On Manpower Scheduling Algorithms. *SIAM Review*, 24 (3), 275–287.



See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/319198823>

# Minimum sum coloring problem: Upper bounds for the chromatic strength

Article in *Discrete Applied Mathematics* · August 2017

DOI: 10.1016/j.dam.2017.07.025

---

CITATIONS

0

---

READS

43

3 authors:



**Clément Lecat**

Université de Picardie Jules Verne

3 PUBLICATIONS 4 CITATIONS

SEE PROFILE



**Corinne Lucet**

Université de Picardie Jules Verne

27 PUBLICATIONS 301 CITATIONS

SEE PROFILE



**Chu Min Li**

Université de Picardie Jules Verne

61 PUBLICATIONS 1,378 CITATIONS

SEE PROFILE

All content following this page was uploaded by [Chu Min Li](#) on 19 October 2017.

The user has requested enhancement of the downloaded file.



# Minimum sum coloring problem: Upper bounds for the chromatic strength

Clément Lecat<sup>\*</sup>, Corinne Lucet, Chu-Min Li

Laboratoire de Modélisation, Information et Système EA 4290, Université de Picardie Jules Verne, Amiens, France

## ARTICLE INFO

### Article history:

Received 14 February 2017  
 Received in revised form 4 July 2017  
 Accepted 20 July 2017  
 Available online 18 August 2017

### Keywords:

Graph theory  
 Sum coloring problem  
 Chromatic strength

## ABSTRACT

The minimum sum coloring problem (*MSCP*) is a vertex coloring problem in which a weight is associated with each color. Its aim is to find a coloring of the vertices of a graph  $G$  with the minimum sum of the weights of the used colors. The *MSCP* has important applications in the fields such as scheduling and VLSI design. The minimum number of colors among all optimal solutions of the *MSCP* for  $G$  is called the *chromatic strength* of  $G$  and is denoted by  $s(G)$ . A tight upper bound of  $s(G)$  allows to significantly reduce the search space when solving the *MSCP*. In this paper, we propose and empirically evaluate two new upper bounds of  $s(G)$  for general graphs,  $UB_A$  and  $UB_S$ , based on an abstraction of all possible colorings of  $G$  formulated as an ordered set of decreasing positive integer sequences. The experimental results on the standard benchmarks DIMACS and COLOR show that  $UB_A$  is competitive, and that  $UB_S$  is significantly tighter than those previously proposed in the literature for 70 graphs out of 74 and, in particular, reaches optimality for 8 graphs. Moreover, both  $UB_A$  and  $UB_S$  can be applied to the more general optimum cost chromatic partition (*OCCP*) problem.

© 2017 Elsevier B.V. All rights reserved.

## 1. Introduction

The optimum cost chromatic partition (*OCCP*) problem [27] consists in finding a valid vertex coloring of a graph  $G = (V, E)$  with some colors  $c_1, c_2, \dots, c_k$ , taking into account their associated weights  $w_1, w_2, \dots, w_k$ , such that the sum of the color weights is minimum. This problem has many applications in VLSI when restricted to circle and permutation graphs [31], in scheduling problems for interval graphs [19] and in resource allocation [1,25].

When  $(w_1, w_2, \dots, w_k) = (1, 2, \dots, k)$ , the *OCCP* problem becomes the minimum sum coloring problem (*MSCP*). The sum of the color weights in an optimal solution of the *MSCP* is denoted by  $\Sigma(G)$ . The *MSCP* was introduced in 1989 and shown to be NP-hard by Kubicka and Schwenk [20]. Several optimal solutions may exist for a graph  $G$ . These optimal solutions give the same  $\Sigma(G)$  but do not necessarily use the same number of colors. The minimum number of colors among all optimal solutions of the *MSCP* for a graph  $G$  is called the *chromatic strength* of  $G$  and is denoted by  $s(G)$ . Note that  $s(G)$  can be bigger than the minimum number  $\chi(G)$  of colors required to color  $G$ , which is called the *chromatic number* of  $G$ .

We find in the literature many theoretical results proving theoretical bounds of  $\Sigma(G)$  or  $s(G)$  [2,13,25,33] and structural properties relative to the graph families for which polynomial algorithms exist [13,19], and a large number of approximation algorithms to solve the *MSCP* for specific graph families (see, e.g., [9,10]). We also find in the literature several heuristic methods, such as the greedy algorithms *MDSAT* and *MRLF* [24], and meta-heuristic algorithms [3,17,28,32] based on evolutionary and tabu search to solve the *MSCP* for general graphs. All of these methods were tested on DIMACS and COLOR

<sup>\*</sup> Corresponding author.

E-mail addresses: [clement.lecat@u-picardie.fr](mailto:clement.lecat@u-picardie.fr) (C. Lecat), [corinne.lucet@u-picardie.fr](mailto:corinne.lucet@u-picardie.fr) (C. Lucet), [chu-min.li@u-picardie.fr](mailto:chu-min.li@u-picardie.fr) (C. Li).

benchmarks [7,15]. Some exact methods were also proposed in [21] based on the classical branch-and-bound scheme or on the minimum/maximum propositional satisfiability (MinSAT/MaxSAT).

When solving the *MSCP* for a general graph  $G$ , the algorithms have to explore the search space, which grows exponentially with the number of colors to be considered. In practice, it is substantially more difficult to reduce this number when solving the *MSCP* than when solving the classical graph coloring problem (*GCP*). In fact, when a valid coloring solution with  $k$  colors is found, the sub-space with  $k$  or more colors can be pruned for the *GCP* because an optimal solution for the *GCP* cannot contain more than  $k$  colors in this case. However, an optimal solution of the *MSCP* can involve more than  $k$  colors. Therefore, the sub-space with  $k$  or more colors cannot be pruned because of the valid coloring solution with  $k$  colors. Consequently, establishing a tight upper bound of  $s(G)$  is essential for solving the *MSCP*.

Different tight bounds of  $s(G)$  have been established in the literature for special graph families, such as trees, interval graphs or partial  $k$ -trees [13,20,27,29]. An upper bound of  $s(G)$  for planar graphs can be derived from a result in [9,10] for the sum multi-coloring problem on planar graphs. Unfortunately, to the best of our knowledge, only two upper bounds of  $s(G)$  have been proposed for a general graph  $G$  in [27] and in [8].

In this paper, we propose two new upper bounds of  $s(G)$  for a general graph  $G$ , called  $UB_A$  and  $UB_S$ . We derive these upper bounds by exploring an abstraction of the set of coloring solutions of  $G$  called *motifs*. Roughly speaking, a motif is a decreasing sequence of positive integers in which each integer represents the number of vertices in  $G$  assigned a color  $c$ . The notion of motifs was already used in [4,5] to solve the *MSCP* for  $P_4$ -sparse graphs and in [22] to compute a lower bound of  $\Sigma(G)$  in the general case. However, to the best of our knowledge, it is the first time that motifs are used for upper bounding  $s(G)$ . By skilfully identifying and excluding those motifs that cannot correspond to an optimal solution of the *MSCP*, we derive the two new upper bounds of  $s(G)$  from the remaining motifs. The experimental results on the standard benchmarks DIMACS and COLOR [6,12] for coloring problems show that  $UB_A$  is competitive and  $UB_S$  is substantially better, in general, than the upper bounds proposed in [8,27]. In particular,  $UB_S$  gives and proves the optimal value of the chromatic strength for eight graphs.

This paper is organized as follows. Section 2 presents the definitions and some well-known graph problems. It also introduces the notions of major colorings and motifs. Section 3 presents the previous upper bounds of  $s(G)$  proposed in the literature and a number of properties of the motifs that allow the new upper bounds  $UB_A$  and  $UB_S$  of  $s(G)$  to be established. Section 4 empirically evaluates  $UB_A$  and  $UB_S$  by comparing them with the previous upper bounds of  $s(G)$  on the DIMACS and COLOR graphs. Section 5 concludes this paper.

## 2. Definitions and formulations

### 2.1. Chromatic number, minimum sum coloring, maximum clique and maximum stable set

We consider an undirected graph  $G = (V, E)$ , where  $V$  is a set of vertices ( $|V| = n$ ) and  $E \subseteq V^2$  is a set of edges. The set of adjacent (or neighbor) vertices of  $v \in V$ , denoted by  $\mathcal{N}$ , is defined as  $\mathcal{N}(v) = \{u \mid (u, v) \in E\}$ . The degree  $d(v)$  of a vertex  $v$  is the number of its adjacent vertices, i.e.,  $d(v) = |\mathcal{N}(v)|$ . The degree of a graph, denoted by  $\Delta(G)$ , is  $\max\{d(v) \mid v \in V\}$ .

A clique  $C$  is a subset of  $V$  such that  $\forall u, v \in C, (u, v) \in E$ . The maximum clique (MaxClique) problem consists in finding a clique with the maximum cardinality in  $G$ .

A stable set  $S$  is a subset of  $V$  such that  $\forall u, v \in S, (u, v) \notin E$ . The maximum stable set (MaxStable) problem consists in finding a stable set with the maximum cardinality in  $G$ .

The complement graph of  $G$  is defined as  $\bar{G} = (V, \bar{E})$ , where  $\bar{E} = V^2 \setminus E$ . Note that a clique in  $G$  is a stable set in  $\bar{G}$  and vice versa.

A coloring of a graph  $G$  with  $k$  colors is a function  $c : V \mapsto \{1, 2, \dots, k\}$  that assigns to each vertex  $v \in V$  a color  $c(v)$ . A coloring is valid if  $\forall (u, v) \in E, c(u) \neq c(v)$ . We denote a valid coloring of  $G$  with  $k$  colors by  $X = \{X_1, X_2, \dots, X_k\}$ , where  $X_i = \{v \in V \mid c(v) = i\}$  is called a color class, that is, a stable set of  $G$ . The graph coloring problem (*GCP*) consists in finding a valid coloring  $X$  of  $G$  with minimum  $k$ . This  $k$  is called the *chromatic number* of  $G$ , and is denoted by  $\chi(G)$ . The *GCP* is NP-hard [18].

The *MSCP* associates a weight  $w_i = i$  with each color  $i$ . We denote  $\Sigma(X)$  as the sum of color weights of a coloring  $X$ :

$$\Sigma(X) = 1 \times |X_1| + 2 \times |X_2| + \dots + k \times |X_k|.$$

**Example 1.** Referring to the graph in Fig. 1,  $X = \{\{a, e\}_1, \{b\}_2, \{c, d, f\}_3\}$  is a valid coloring. The vertices  $a$  and  $e$  are colored with color 1, the vertex  $b$  with color 2, and the vertices  $c, d$  and  $f$  with color 3. The sum coloring is  $\Sigma(X) = 1 \times 2 + 2 \times 1 + 3 \times 3 = 13$ .

Given a graph  $G$ , the *MSCP* consists in finding a valid coloring  $X$  of  $G$  with the minimum sum of color weights. This minimum sum is called the chromatic sum of  $G$  and is denoted by  $\Sigma(G)$ :

$$\Sigma(G) = \min\{\Sigma(X) \mid X \text{ is a valid coloring of } G\}$$

Kubicka and Schwenk proved that the *MSCP* is NP-hard [20]. An optimal solution of the *GCP* does not necessarily correspond to an optimal solution of the *MSCP*. For example, an optimal solution of the *GCP* for the graph in Fig. 2 uses 2 colors and  $\Sigma(X) = 12$ , while an optimal solution of the *MSCP* for this graph uses 3 colors. The chromatic sum of the graph is

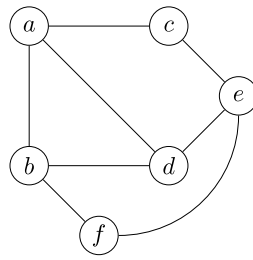


Fig. 1. A graph.

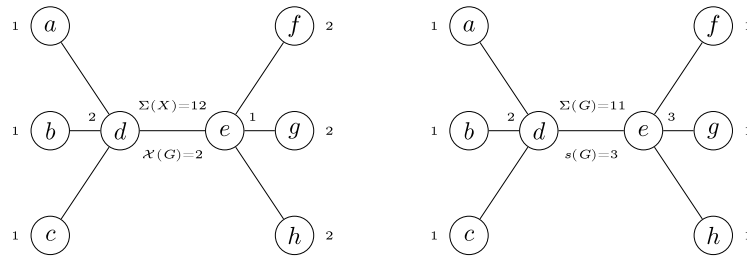


Fig. 2. An optimal solution of the GCP,  $X = \{\{a, b, c, e\}_1, \{d, f, g, h\}_2\}$  with  $\Sigma(X) = 12$ , and an optimal solution of the MSCP,  $X' = \{\{a, b, c, f, g, h\}_1, \{d\}_2, \{e\}_3\}$  with  $\Sigma(X') = 11$ .

11. If the colors are dates and the vertices of the graph are tasks (e.g., task  $d$  is completed at date 2), the average completion time provided by the MSCP (e.g.,  $11/8 = 1.375$ ) is less than or equal to that given by the GCP (e.g.,  $12/8 = 1.5$ ).

The minimum number of colors among all optimal solutions of the MSCP for a graph  $G$  is called the *chromatic strength* (or simply the *strength*) of  $G$ , and is denoted by  $s(G)$ .

For graph coloring problems such as the GCP or MSCP, a valid coloring of a graph  $G = (V, E)$  with  $k$  colors must be found in the search space of cardinality  $k^{|V|}$ . The number of considered colors,  $k$ , must be sufficiently large to achieve an optimal solution. While the GCP and MSCP are both NP-hard, the MSCP is substantially more difficult to solve than the GCP in practice because it is considerably more complex to prune a search space when solving the MSCP. Indeed, any valid coloring with  $k$  colors provides an upper bound for  $\chi(G)$ , and then, the sub-space with more than  $k - 1$  colors can be reduced when solving the GCP. However, in the case of the MSCP,  $k' > k$  colors could be necessary to reach an optimal solution. Therefore, such reductions are impossible. Consequently, it is essential to determine as tight as possible an upper bound of  $s(G)$  for solving the MSCP.

### 2.2. Major coloring

For a valid coloring  $X$ , each color class  $X_i$  is a stable set. We can exchange two colors,  $i$  and  $j$ , without affecting the validity of  $X$ . The set of colorings that can be achieved by such an exchange in  $X$  forms an equivalence class denoted by  $\Psi(X)$  of symmetric colorings. All colorings of  $\Psi(X)$  use the same number of colors, but they do not provide the same sum of color weights. Referring to Fig. 1, the sum of color weights of the coloring  $\{\{a, e\}_1, \{b\}_2, \{c, d, f\}_3\}$  is 13, whereas the sum of the color weights of the coloring  $\{\{a, e\}_1, \{c, d, f\}_2, \{b\}_3\}$  is 11. Therefore, we define the notion of *major coloring*.

**Definition 1.** A major coloring of  $\Psi(X)$ , denoted by  $X^m$ , is a coloring  $X^m = \{X_1, X_2, \dots, X_k\}$  such that  $|X_1| \geq |X_2| \geq \dots \geq |X_k|$ .

**Example 2.** Referring to Fig. 1, the coloring  $X = \{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is a major coloring of the graph.  $\Sigma(X^m) = 10$ .

The following property is a direct consequence of Definition 1.

**Property 1.** Let  $\Psi(X)$  be the set of symmetric colorings of  $X$ ; then,  $\forall X' \in \Psi(X), \Sigma(X^m) \leq \Sigma(X')$ .

Consequently, only major colorings need to be considered when solving the MSCP. In the following, all colorings we consider are major and are simply written as  $X$ .

### 2.3. Motifs

Any major coloring  $X$  with  $k$  colors corresponds to a decreasing sequence  $p$  of positive integers  $(|X_1|, |X_2|, \dots, |X_k|)$ , called the *motif* of  $X$ . The  $i$ th integer is denoted by  $p[i] = |X_i|$ . The number of integers in  $p$  is denoted by  $|p|$ . The sum of color weights

of  $X$  is computed by Eq. (1).

$$\Sigma(X) = \Sigma(p) = 1 \times p[1] + 2 \times p[2] + \dots + k \times p[k]. \tag{1}$$

**Example 3.** The motif corresponding to  $X = \{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is  $p = (3, 2, 1)$ , with  $p[1] = 3, p[2] = 2$  and  $p[3] = 1$ .  $\Sigma(X) = \Sigma(p) = 10$ .

The motifs are of interest to the *MSCP* because each motif is an abstraction of several colorings sharing the same weight sum. Therefore, investigating the set of motifs instead of the set of major colorings substantially reduces the search space. For example, referring to Fig. 1, the two different colorings  $\{\{c, d, f\}_1, \{b, e\}_2, \{a\}_3\}$  and  $\{\{a, e, f\}_1, \{b, c\}_2, \{e\}_3\}$  correspond to the same motif  $p = (3, 2, 1)$ . Two different motifs correspond necessarily to different colorings. Nevertheless, a motif can correspond to an invalid coloring because it does not include any structural property of a graph. For example, referring to Fig. 1, the motifs  $(5, 1)$  and  $(4, 1, 1)$  do not represent any valid coloring of the graph. As shown in Section 3.4, some characteristics of a graph can be used to exclude a part of the motifs that correspond to invalid colorings, so that we can derive interesting properties for the *MSCP* from the remaining motifs.

For any graph with  $n$  vertices, we denote by  $\phi(n)$  the set of motifs, and by  $\phi(n, k)$  the set of all motifs with  $k$  colors.

$$\phi(n, k) = \{p \in \phi(n) \mid |p| = k\}$$

$$\phi(n) = \bigcup_{k=1}^n \{\phi(n, k)\}$$

The number of motifs in  $\phi(n)$  is equal to the number of partitions of  $n$  into decreasing positive integers, which grows exponentially with  $n$ . Hardy and Ramanujan [11] gave an approximate number of partitions of  $n$  into decreasing positive integers:

$$|\phi(n)| \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{2n/3}}. \tag{2}$$

Our approach operates in  $\phi(n)$  to find a tight upper bound of the chromatic strength of any graph with  $n$  vertices, using the dominance relation between the motifs defined in the following subsection. As shown later, we do not need to enumerate the entire set of motifs, but only a very small subset of appropriate motifs.

#### 2.4. Dominance relation

The dominance relation, denoted by  $\succeq$ , between the motifs of  $\phi(n)$  was introduced by Bonomo and Valencia-Pabon [4,5] to compute the chromatic sum for a subset of  $P_4$ -sparse graphs or an upper bound of the chromatic sum for general  $P_4$ -sparse graphs. We adapt their definition in our approach to derive tight upper bounds of the chromatic strength for general graphs.

**Definition 2.** Let  $p$  and  $q$  be two motifs in  $\phi(n)$ . We state that  $p$  dominates  $q$ , denoted by  $p \succeq q$ , if and only if  $\forall t$  such that  $1 \leq t \leq \min\{|p|, |q|\}, \sum_{x=1}^t p[x] \geq \sum_{x=1}^t q[x]$ .

The main difference between Definition 2 and the original definition in [4] is that we explicitly require  $\sum_{x=1}^{|p|} p[x] = \sum_{x=1}^{|q|} q[x] = n$  because  $p$  and  $q$  are both in  $\phi(n)$ .

Two motifs are not necessarily comparable, as shown in Example 4.

**Example 4.** Let  $p$  and  $q$  be two motifs of a graph  $G$  with 15 vertices,  $p = (9, 3, 3)$  and  $q = (8, 6, 1)$ .

When  $t = 1, \sum_{i=1}^1 p[i] > \sum_{i=1}^1 q[i] (9 > 8)$ .

When  $t = 2, \sum_{i=1}^2 p[i] < \sum_{i=1}^2 q[i] (9 + 3 < 8 + 6)$ .

Therefore,  $p \not\succeq q$  and  $q \not\succeq p$ , i.e.,  $p$  and  $q$  are not comparable.

In other words, the dominance relation  $\succeq$  is a partial order. The dominance relation is of interest to the *MSCP* because of the following property.

**Property 2.** Let  $G$  be a graph,  $c_1, c_2, \dots, c_k$  be  $k$  colors and  $w_i$  be the weight associated with  $c_i (1 \leq i \leq k)$  such that  $w_1 \leq w_2 \leq \dots \leq w_k$ . Let  $p$  and  $q$  be two motifs corresponding respectively to two valid colorings,  $X$  and  $X'$ , of  $G$ . If  $p \succeq q$ , then  $\sum_{i=1}^{|p|} w_i |X_i| \leq \sum_{i=1}^{|q|} w_i |X'_i|$ .

**Proof.** Note that  $p \succeq q$  implies  $|p| \leq |q|$  because if  $|p| > |q|$ , then  $n = \sum_{i=1}^{|p|} p[i] > \sum_{i=1}^{|q|} p[i] \geq \sum_{i=1}^{|q|} q[i] = n$  according to Definition 2, which is impossible. Therefore, we can re-write  $\sum_{i=1}^{|p|} w_i |X_i| - \sum_{i=1}^{|q|} w_i |X'_i|$  as  $\sum_{i=1}^{|p|} w_i (p[i] - q[i]) - \sum_{i=|p|+1}^{|q|} w_i q[i]$ . Since  $\forall t$  such that  $1 \leq t \leq |p|, \sum_{i=1}^t p[i] \geq \sum_{i=1}^t q[i]$ , we have  $\sum_{i=1}^{|p|} w_i (p[i] - q[i]) \leq w_2(p[1] - q[1]) + \sum_{i=2}^{|p|} w_i (p[i] - q[i]) = w_2(p[1] + p[2] - q[1] - q[2]) + \sum_{i=3}^{|p|} w_i (p[i] - q[i]) \leq w_3(p[1] + p[2] - q[1] - q[2]) + \sum_{i=3}^{|p|} w_i (p[i] - q[i]) = w_3(p[1] + p[2] + p[3] - q[1] - q[2] - q[3]) + \sum_{i=4}^{|p|} w_i (p[i] - q[i]) \leq \dots \leq w_{|p|} (\sum_{i=1}^{|p|} p[i] - \sum_{i=1}^{|p|} q[i])$ . Each of the above inequalities holds because

we have replaced  $w_t$  in  $w_t(\sum_{i=1}^t p[i] - \sum_{i=1}^t q[i])$  with  $w_{t+1}(\geq w_t)$ , which gives a greater number  $w_{t+1}(\sum_{i=1}^t p[i] - \sum_{i=1}^t q[i])$  because  $\sum_{i=1}^t p[i] - \sum_{i=1}^t q[i] \geq 0$ .

Since  $\sum_{i=1}^{|p|} p[i] - \sum_{i=1}^{|p|} q[i] = n - \sum_{i=1}^{|p|} q[i] = \sum_{i=|p|+1}^{|q|} q[i]$ , we have  $\sum_{i=1}^{|p|} w_i |X_i| - \sum_{i=1}^{|q|} w_i |X'_i| \leq w_{|p|} \sum_{i=|p|+1}^{|q|} q[i] - \sum_{i=|p|+1}^{|q|} w_i q[i] \leq w_{|p|} \sum_{i=|p|+1}^{|q|} q[i] - \sum_{i=|p|+1}^{|q|} w_{|p|} q[i] = 0$  because  $w_i \geq w_{|p|}$  if  $i > |p|$ .

Therefore,  $\sum_{i=1}^{|p|} w_i |X_i| = \sum_{i=1}^{|p|} w_i p[i] \leq \sum_{i=1}^{|q|} w_i q[i] = \sum_{i=1}^{|q|} w_i |X'_i|$ .  $\square$

**Property 2** generalizes a property for the *MSCP* presented in [4,5], in which  $(w_1, w_2, \dots, w_k) = (1, 2, \dots, k)$ , to the *OCCP* problem.

### 3. Tight upper bounds for the chromatic strength of a graph

In studies in the literature, different relationships have been established between the structural properties of a graph  $G$  (degree, chromatic number) and its chromatic sum [2,5,14,30], but few structural properties have been utilized to bound the chromatic strength for general graphs. **Property 3** states a trivial upper bound of  $s(G)$  given in [27]:

**Property 3.** Let  $G$  be a graph and  $\Delta(G)$  its degree; then,

$$s(G) \leq \Delta(G) + 1.$$

A better upper bound based on a valid coloring of  $G$  is given in [8] and is stated in **Property 4**.

**Property 4.** Let  $G$  be a graph,  $\chi(G)$  its chromatic number and  $X$  a valid coloring with  $k$  colors; then,

$$s(G) \leq \left\lceil \frac{\Delta(G) + \chi(G)}{2} \right\rceil \leq \left\lceil \frac{\Delta(G) + k}{2} \right\rceil.$$

Other bounds of  $s(G)$  in our knowledge are for special graph families. **Property 5** [20] and **Property 6** [27] both concern the tree family.

**Property 5.** Let  $T$  be a tree with  $n$  vertices; then,  $s(T)$  is bounded by  $\Omega(\log(n))$ .

**Property 6.** Let  $T$  be a tree and  $\rho$  be the number of vertices on the longest path in  $T$ ; then,

$$s(T) \leq 1 + \left\lceil \frac{\rho}{2} \right\rceil.$$

**Property 7** is related to partial  $k$ -trees [13] and **Property 8** to arbitrary interval graphs [29].

**Property 7.** Let  $G$  be a partial  $k$ -tree with  $n$  vertices; then,  $s(G)$  is bounded by  $\Omega(k \times \log(n))$ .

**Property 8.** Let  $G$  be an arbitrary interval graph and  $\chi(G)$  its chromatic number; then,

$$s(G) \leq 2 \times \chi(G) - 1.$$

**Property 9** gives an upper bound of the number of colors in an optimal solution of the sum multi-coloring problem on planar graphs [9,10].

**Property 9.** Let  $G = (V, E)$  be a planar graph with  $n$  vertices, in which any vertex  $v$  should be colored using a set  $p(v)$  of colors, and for any adjacent vertices  $u$  and  $v$ ,  $p(u)$  and  $p(v)$  should be disjoint. Let  $p_{max} = \max\{|p(v)| \mid v \in V\}$ . The total number of colors in a multi-coloring of  $G$  minimizing the sum of the weights of the used colors is bounded by  $O(p_{max} \times \chi(G) \times \log(n))$ .

A direct consequence of **Property 9** is that  $s(G)$  is bounded by  $O(\chi(G) \times \log(n))$  if  $G$  is a planar graph, because  $p_{max} = 1$  in the *MSCP*.

In this section, we propose two new upper bounds,  $UB_A$  and  $UB_S$ , of  $s(G)$  for a general graph  $G$ . They are based on a valid coloring and an exploration of the set of motifs  $\phi(n)$ . We first define a total order in  $\phi(n)$  and propose an algorithm that allows to assign to each motif in  $\phi(n, k)$  an index, which is useful to understand the construction principle of the new upper bounds, and then, we present  $UB_A$  and  $UB_S$ .

#### 3.1. A total order in $\phi(n)$

The set of motifs  $\phi(n)$  is first sorted in ascending order of the number of colors:  $\phi(n, 1), \phi(n, 2), \dots, \phi(n, n)$ . Then, each  $\phi(n, k)$  is sorted in decreasing lexicographical order. Let  $p$  and  $q$  be two motifs in  $\phi(n, k)$ ;  $p$  is lexicographically greater than  $q$  if  $p[1] > q[1]$  or  $\exists t \ p[t] = q[t]$  for  $1 \leq i < t$  and  $p[t] > q[t]$ . As an example, **Table 1** lists all the motifs in  $\phi(8)$  in the above defined order.

**Table 1**  
 $\phi(8)$ .

$\phi(n, k)$	$p \in \phi(n, k)$	$\Sigma(p)$
$\phi(8, 1)$	(8)	8
	(7, 1)	9
$\phi(8, 2)$	(6, 2)	10
	(5, 3)	11
	(4, 4)	12
	(6, 1, 1)	11
$\phi(8, 3)$	(5, 2, 1)	12
	(4, 3, 1)	13
	(4, 2, 2)	14
	(3, 3, 2)	15
	(5, 1, 1, 1)	14
$\phi(8, 4)$	(4, 2, 1, 1)	15
	(3, 3, 1, 1)	16
	(3, 2, 2, 1)	17
	(2, 2, 2, 2)	20
$\phi(8, 5)$	(4, 1, 1, 1, 1)	18
	(3, 2, 1, 1, 1)	19
	(2, 2, 2, 1, 1)	21
$\phi(8, 6)$	(3, 1, 1, 1, 1, 1)	23
	(2, 2, 1, 1, 1, 1)	24
$\phi(8, 7)$	(2, 1, 1, 1, 1, 1, 1)	29
$\phi(8, 8)$	(1, 1, 1, 1, 1, 1, 1, 1)	36

**Definition 3.** We denote by  $p_k^i$  the  $i$ th motif in the decreasing lexicographic order in  $\phi(n, k)$ . Let  $p_k^i$  and  $p_{k'}^j$  be two motifs in  $\phi(n)$ . We state that  $p_k^i$  precedes (or is greater than)  $p_{k'}^j$  in  $\phi(n)$  if  $k < k'$  or  $k = k'$  and  $i < j$ .

The first motif  $p_k^1$  in  $\phi(n, k)$  is

$$(n - k + 1, \overbrace{1, 1, \dots, 1}^{k-1 \text{ times}})$$

and the last motif in  $\phi(n, k)$  is

$$\left( \overbrace{\left\lceil \frac{n}{k} \right\rceil, \dots, \left\lceil \frac{n}{k} \right\rceil}^{n \bmod k \text{ times}}, \overbrace{\left\lfloor \frac{n}{k} \right\rfloor, \dots, \left\lfloor \frac{n}{k} \right\rfloor}^{k - n \bmod k \text{ times}} \right)$$

In fact, we have the following property.

**Property 10.** For any motif  $p_k^i \in \phi(n, k)$ , it holds that  $\left\lceil \frac{n}{k} \right\rceil \leq p_k^i[1] \leq n - k + 1$ .

**Proof.** Let  $\lambda = p_k^i[1]$ . Note that  $p_k^i$  contains  $k$  positive integers in the decreasing order, of which the sum is  $n$ . If  $\left\lceil \frac{n}{k} \right\rceil > \lambda$ , then  $n = \sum_{x=1}^k p_k^i[x] \leq \lambda \times k \leq (\left\lceil \frac{n}{k} \right\rceil - 1) \times k < n$ , which is impossible. If  $\lambda > n - k + 1$ , then  $n = \sum_{x=1}^k p_k^i[x] \geq \lambda + k - 1 > n$ , which again is impossible.  $\square$

Algorithm 1 enumerates the motifs in the set  $\phi(n, k)$  in decreasing lexicographic order. The parameters are integers  $n$  and  $k$  such that  $n \geq k$ , a motif  $p$  under construction (initially empty) and a resulting set  $\phi$  of motifs. Each partition begins by applying the first integer, which should be between  $sup$  and  $\left\lceil \frac{n}{k} \right\rceil$ , where  $sup$  is equal to  $min(n - k + 1, min(p))$ , so that the decreasing order is respected during the recursive process. At the end, the parameter  $\phi = \phi(n, k)$ .

For example, in the call MOTIF(8, 3,  $\emptyset, \emptyset$ ) ( $p=\phi=\emptyset$ ), the algorithm generates each first integer between 6 and 3. When the first integer is 6, the algorithm calls MOTIF(2, 2, {6},  $\emptyset$ ), which generates only one partition of 2 into 2 integers (i.e., (1, 1)) appended to {6} to form a complete motif (6, 1, 1). When the first integer is 4, the algorithm calls MOTIF(4, 2, {4}, {(6, 1, 1), (5, 3, 2)}), which generates two partitions of 4 into 2 integers, (3, 1) and (2, 2), each partition being appended to {4} to form a complete motif. See Table 1 for all the results.



---

**Algorithm 1** : MOTIF( $n, k, p, \phi$ )

---

**Input** : two positive integers  $n$  and  $k$  such that  $n \geq k$ , motif  $p$  under construction, set of motifs  $\phi$  under construction

**Output** : a set of motifs  $\phi$  in decreasing lexicographic order

```

1 begin
2   if  $k = 1$  then
3      $\phi \leftarrow \phi \cup \{p \cup \{n\}\}$ ;
4   else
5     if  $p = \emptyset$  then
6        $sup \leftarrow n - k + 1$ ;
7     else
8        $sup \leftarrow \min(n - k + 1, \min(p))$ ;
9      $inf \leftarrow \lceil \frac{n}{k} \rceil$ ;
10    foreach  $x \leftarrow sup$  downto  $inf$  do
11      Motif( $n - x, k - 1, p \cup \{x\}, \phi$ );

```

---

### 3.2. Construction principle of the new upper bounds

Given a valid coloring  $X$  of a graph  $G$  and its associated motif  $p$ , our objective is to determine the number  $k_t \in [1..n]$  of colors such that all motifs with  $k_t$  or more colors are dominated by  $p$ , as specified by Eq. (3).

$$\forall q \in \mathcal{U}_{k_t} = \bigcup_{x=k_t}^n \phi(n, x), p \succeq q \tag{3}$$

Note that  $\mathcal{U}_{k_t}$  in Eq. (3) is not empty because it trivially contains the only motif  $(1, 1, \dots, 1)$  in  $\phi(n, n)$  for  $k_t = n$ . Such a  $\mathcal{U}_{k_t}$  does not contain any valid coloring better than  $X$ . Therefore,  $k_t$  is an upper bound of  $s(G)$ .

Based on this principle, we establish two upper bounds for  $s(G)$  in the following two subsections:  $UB_A$ , an algebraic upper bound, and  $UB_S$ , an algorithmic upper bound based on a maximum stable set of  $G$ .

### 3.3. $UB_A$ , an algebraic upper bound for $s(G)$

$UB_A$  is based on the following two properties.

**Property 11.** Let  $G = (V, E)$  with  $|V| = n$  and  $k \leq n$  be an integer. The motif  $p_k^1 = (n - k + 1, 1, 1, \dots, 1)$  dominates all other motifs in  $\phi(n, k)$ .

In fact, let  $q \in \phi(n, k)$ . Then,  $\forall t$  such that  $1 \leq t \leq k$ ,  $\sum_{x=t+1}^k p_k^1[x] \leq \sum_{x=t+1}^k q[x]$  because  $p_k^1[x] = 1 \leq q[x]$  when  $x > 1$ . Therefore,  $\sum_{x=1}^t p_k^1[x] = n - \sum_{x=t+1}^k p_k^1[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ .

**Property 12.** Let  $G = (V, E)$  with  $|V| = n$  and  $k$  and  $k'$  be two integers such that  $1 \leq k < k' \leq n$ . The motif  $p_k^1 = (n - k + 1, 1, 1, \dots, 1)$  dominates  $p_{k'}^1 = (n - k' + 1, 1, 1, \dots, 1)$ .

In fact, since  $n - k + 1 > n - k' + 1$ ,  $\forall t$  such that  $1 \leq t \leq k$ ,  $\sum_{x=1}^t p_k^1[x] \geq \sum_{x=1}^t p_{k'}^1[x]$ .

Properties 11 and 12 mean that, if  $k \leq k'$ , then the motif  $p_k^1 = (n - k + 1, 1, 1, \dots, 1)$  dominates all motifs in  $\phi(n, k')$ . The sum coloring of  $p_k^1$  is

$$\Sigma(p_k^1) = (n - k + 1) + \sum_{x=2}^k x = \frac{1}{2}k^2 - \frac{1}{2}k + n \tag{4}$$

Given a valid coloring, let  $k_t$  be the smallest number  $k$  of colors such that

$$\Sigma(p_{k_t}^1) \geq \Sigma(X) \tag{5}$$

Eq. (5) signifies that a sum coloring with  $k_t$  or more colors cannot be better than the known valid coloring  $X$  according to Properties 11 and 12. Therefore, the optimal solution must use at most  $k_t - 1$  colors if  $X$  uses fewer than  $k_t$  colors; otherwise  $k_t$ .

Using Eqs. (4), (5) is transformed to

$$\frac{1}{2}k_t^2 - \frac{1}{2}k_t + n - \Sigma(X) \geq 0 \tag{6}$$



Eq. (6) is valid if and only if

$$k_t \geq \frac{1 + \sqrt{1 + 8 \times (\Sigma(X) - n)}}{2}$$

The new algebraic upper bound  $UB_A$  of  $s(G)$  is defined in Eq. (7).

$$UB_A = \max \left( \left\lceil \frac{1 + \sqrt{1 + 8 \times (\Sigma(X) - n)}}{2} \right\rceil - 1, |X| \right) \tag{7}$$

Note that, other than the valid coloring  $X$ ,  $UB_A$  does not consider any structural information of  $G$ . We show in the next subsection that we can obtain a tighter upper bound by taking into account more structural information of  $G$ , i.e., its maximum stable set cardinality.

### 3.4. $UB_S$ , an algorithmic upper bound of $s(G)$

Although the MaxClique and the GCP and MSCP are all NP-hard problems, the MaxClique problem is easier to solve than the GCP and MSCP in practice. For example, the state-of-the-art exact algorithm IncMaxCLQ [23] for MaxClique solves (i.e., finds and proves an optimal solution of) any random graph of 200 vertices with any density in a few seconds (a random graph of density  $d$  is generated by making each pair of vertices adjacent with probability  $d$ ). However, to the best of our knowledge no exact algorithm exists that can find the chromatic number or the chromatic sum of a random graph of 200 vertices with density 0.5 in reasonable time. As mentioned in Section 2, the maximum clique of  $\bar{G}$  is a maximum stable set of  $G$ . Our new algorithmic upper bound,  $UB_S$ , is based on a maximum clique of  $\bar{G}$  found by IncMaxCLQ, based on Property 13.

**Property 13.** Let  $\alpha(G)$  denote the cardinality of a maximum stable set of  $G$ . Any motif  $p$  for which  $p[1] > \alpha(G)$  cannot correspond to a valid coloring of  $G$ .

In consequence, all motifs in which the first integer is greater than  $\alpha(G)$  can be excluded from consideration. Further exclusion can be conducted based on the following observation. Given any valid major coloring  $X$ , it could be possible to find two color classes  $X_i$  and  $X_j$  ( $i < j$ ) such that a vertex of  $X_j$  can be moved into  $X_i$  to obtain another valid major coloring,  $X'$ . Clearly,  $\Sigma(X') < \Sigma(X)$  and the motif  $p'$  associated with  $X'$  dominates the motif  $p$  associated with  $X$ . The motif  $p'$  is obtained by decrementing the  $j$ th integer of  $p$  by 1 and incrementing the  $i$ th integer of  $p$  by 1. We call this transformation of  $p$  the left-shifting operation. Note that this operation is not always possible because one must keep the integers of the resulting motif in decreasing order. Moreover, all integers in the resulting motif should be positive. Given two integers,  $n$  and  $k$ , we are interested in those motifs in  $\phi(n, k)$ , called the major motifs, that cannot be transformed into another motif in  $\phi(n, k)$  by a left-shifting operation without incrementing the first integer.

Let  $\lambda$  denote the first integer of a motif  $p$  in  $\phi(n, k)$ . According to Property 10,  $\lceil \frac{n}{k} \rceil \leq \lambda \leq n - k + 1$ . Intuitively, a motif is major if it contains the maximum number ( $\beta$ ) of integers equal to its first integer,  $\lambda$ , that cannot be incremented without incrementing the first integer. The remaining value of  $n$ , i.e.,  $n - \beta \times \lambda$ , should be partitioned into  $k - \beta$  positive integers. Therefore,  $\beta$  is the maximum integer satisfying  $n - \beta \times \lambda \geq k - \beta$  or  $\beta \leq \frac{n-k}{\lambda-1}$  after excluding the trivial motif  $(1, 1, \dots, 1)$  and assuming  $\lambda > 1$ . Therefore, we set  $\beta$  to  $\lfloor \frac{n-k}{\lambda-1} \rfloor$ .

A major motif is formally defined in Definition 4.

**Definition 4.** Let  $\lambda$  and  $\beta$  be two integers such that  $\lceil \frac{n}{k} \rceil \leq \lambda \leq n - k + 1$  and  $\beta = \lfloor \frac{n-k}{\lambda-1} \rfloor$ . A major motif in  $\phi(n, k)$  is a motif  $p_k^i$  with the properties

$$\begin{cases} p_k^i[x] = \lambda, & \text{if } 1 \leq x \leq \beta; \\ p_k^i[x] = n - \beta \times \lambda - (k - \beta - 1), & \text{if } x = \beta + 1; \\ p_k^i[x] = 1, & \text{if } \beta + 1 < x \leq k. \end{cases}$$

**Example 5.** Consider  $\phi(8, 4)$  (referring to Table 1). The major motifs are  $(5, 1, 1, 1)$ ,  $(4, 2, 1, 1)$ ,  $(3, 3, 1, 1)$  and  $(2, 2, 2, 2)$ . Indeed, none of these motifs can be transformed by a left-shifting operation without increasing the first integer. For motif  $(4, 2, 1, 1)$ ,  $\lambda = 4$ ,  $\beta = 1$  and for motif  $(3, 3, 1, 1)$ ,  $\lambda = 3$ ,  $\beta = 2$ .

The major motif notion is of interest because of the following two properties. The first, Property 14, states that a major motif in  $\phi(n, k)$  dominates all motifs  $\phi(n, k)$  with the same first integer and represents the smallest sum coloring among these motifs.

**Property 14.** Let  $p$  and  $q$  be two motifs in  $\phi(n, k)$  such that  $p$  is major and  $p[1] = q[1]$ ; then,  $p \geq q$ .

**Proof.** Let  $\beta = \lfloor \frac{n-k}{p[1]-1} \rfloor$ . Since  $p$  is major, we have  $p[1] = p[2] = \dots = p[\beta] = q[1] \geq q[2] \geq \dots \geq q[\beta]$ . Therefore,  $\forall t$  such that  $1 \leq t \leq \beta$ ,  $\sum_{x=1}^t p[x] \geq \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta + 1 \leq t \leq k$ , we have  $p[t + 1] = p[t + 2] = \dots = p[k] = 1 \leq q[k] \leq q[k - 1] \leq \dots \leq q[t + 1]$ . Therefore,  $\sum_{x=1}^t p[x] = n - \sum_{x=t+1}^k p[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ .

Therefore,  $p \geq q$ .  $\square$

Property 15 compares two major motifs in  $\phi(n, k)$ .

**Property 15.** Let  $p$  and  $q$  be two major motifs in  $\phi(n, k)$ . If  $p[1] > q[1]$ , then  $p \succeq q$ .

**Proof.** Let  $\beta_p = \lfloor \frac{n-k}{p[1]-1} \rfloor$  and  $\beta_q = \lfloor \frac{n-k}{q[1]-1} \rfloor$ . Clearly  $\beta_p < \beta_q$ . We have  $p[1] = p[2] = \dots = p[\beta_p] > q[1] = q[2] = \dots = q[\beta_p]$ . Therefore,  $\forall t$  such that  $1 \leq t \leq \beta_p$ ,  $\sum_{x=1}^t p[x] > \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta_p + 1 \leq t \leq k$ , we have  $p[t+1] = p[t+2] = \dots = p[k] = 1 \leq q[k] \leq q[k-1] \dots \leq q[t+1]$ , implying  $\sum_{x=1}^t p[x] = n - \sum_{x=t+1}^k p[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ . Therefore,  $p \succeq q$ .  $\square$

Since the motifs in  $\phi(n, k)$  are in decreasing lexicographical order, Properties 14 and 15 imply that a major motif in  $\phi(n, k)$  dominates all its successors in  $\phi(n, k)$ , as stated in Property 16.

**Property 16.** Let  $p_k^i \in \phi(n, k)$  be a major motif and  $p_k^j \in \phi(n, k)$  such that  $j > i$ . Then,  $p_k^i \succeq p_k^j$ .

Observe that while a major coloring is the best sum coloring in a set of symmetric colorings, a major motif is the best motif in a set of motifs in the sense of Property 16. A direct consequence is that  $\forall p_k^i \in \phi(n, k)$ ,  $p_k^1 \succeq p_k^i$  because  $p_k^1$  is major and is the first motif in  $\phi(n, k)$ .

The following property compares the major motif  $p_k^1$  to  $p_{k'}^i$  when  $k < k'$ .

**Property 17.** Let  $k$  and  $k'$  be two integers such that  $k < k'$ , and  $\mathcal{U}_{k'} = \bigcup_{x=k'}^n \{\phi(n, x)\}$ ; then,  $\forall q \in \mathcal{U}_{k'}$ ,  $p_k^1 \succeq q$ .

**Proof.** According to Property 12,  $p_k^1$  dominates  $p_x^1 \forall x$  such that  $k' \leq x \leq n$ . Since  $p_x^1$  dominates all motifs in  $\phi(n, x)$  according to Property 16,  $p_k^1$  dominates all motifs in  $\mathcal{U}_{k'}$ .  $\square$

The following property is the main idea of the algorithmic upper bound proposed in this paper. It establishes that any major motif  $p$  of size  $k$  dominates all motifs of larger size and smaller or equal first integer.

**Property 18.** Let  $k$  and  $k'$  be two integers such that  $k < k'$ . Let  $p_k^i$  be a major motif and  $\mathcal{U}_{k'} = \bigcup_{y=k'}^n \{\phi(n, y)\}$ . Then,  $\forall q \in \mathcal{U}_{k'}$  such that  $q[1] \leq p_k^i[1]$ ,  $p_k^i \succeq q$ .

**Proof.** Let  $\beta = \lfloor \frac{n-k}{p_k^i[1]-1} \rfloor$ . Since  $p_k^i$  is major, we have  $p_k^i[1] = p_k^i[2] = \dots = p_k^i[\beta] \geq q[1] \geq q[2] \geq \dots \geq q[\beta]$ . Therefore,  $\forall t$  such that  $1 \leq t \leq \beta$ ,  $\sum_{x=1}^t p_k^i[x] \geq \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta + 1 \leq t \leq k$ , we have  $p_k^i[t+1] = p_k^i[t+2] = \dots = p_k^i[k] = 1 \leq q[k] \leq q[k-1] \leq \dots \leq q[t+1]$  ( $q$  is a sequence of decreasing positive integers), implying  $\sum_{x=1}^t p_k^i[x] = n - \sum_{x=t+1}^k p_k^i[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ . Therefore,  $p_k^i \succeq q$ .  $\square$

Given a valid coloring  $X$  of a graph  $G$  with  $n$  vertices and the cardinality  $\alpha(G)$  of a maximum stable set of  $G$ , we can search for the smallest  $k$  and the major motif  $p_k^i$  with  $p_k^i[1] = \alpha(G)$  such that  $\Sigma(X) < \Sigma(p_k^i)$ . Property 18 states that a valid coloring of  $G$  with  $k$  or more colors better than  $X$  cannot be found. Therefore, we define  $UB_S$  as  $k - 1$ . Algorithm 2 implements  $UB_S$ . The function *constructMajorMotif*( $\lambda, k$ ) constructs a major motif  $p \in \phi(n, k)$  such that  $p[1] = \lambda$ , as defined in Definition 4. The function *computeSumcoloring*( $p$ ) computes the sum of color weights of  $p$ , according to Eq. (1).

---

**Algorithm 2 :**  $CUB_S(n, \alpha^*, X)$

---

**Input** : the number of vertices  $n$  in a graph  $G$ , the cardinality (or an upper bound of the cardinality)  $\alpha^*$  of a maximum stable set in  $G$ , and a valid coloring  $X$  for  $G$

**Output** : an upper bound of  $s(G)$

```

1 begin
2    $SUM \leftarrow 0$ ;
3    $k \leftarrow |X|$ ; /*initialize  $k$ */
4    $\lambda \leftarrow \alpha^*$ ;
5   while  $SUM \leq \Sigma(X)$  do
6      $k \leftarrow k + 1$ ;
7     if  $\lambda > n - k + 1$  then
8        $\lambda \leftarrow n - k + 1$ ;
9      $p \leftarrow \text{constructMajorMotif}(\lambda, k)$ ;
10     $SUM \leftarrow \text{computeSumcoloring}(p)$ ;
11  return  $k - 1$ ;

```

---

The complexity of both the *constructMajorMotif*( $\lambda, k$ ) and *computeSumcoloring*( $p$ ) functions is  $O(k)$ . Since  $k \leq n$ , the complexity of Algorithm 2 is  $O(n^2)$ .

We note that the different properties and bounds concerning the *MSCP* are valid for the *OCCP* problem according to [Property 2](#).

#### 4. Empirical evaluation

In this section, we compare our algebraic and algorithmic upper bounds ( $UB_A$  and  $UB_S$ ) of the chromatic strength of a general graph  $G$  with the upper bound  $UB_{hmt}$  proposed by Hajiabolhassan et al. [8]; see [Property 4](#). To find a maximum stable set of  $G$ , we run the state-of-the-art exact MaxClique algorithm, IncMaxCLQ [23] to find a maximum clique in the complement graph  $\bar{G}$ . Although the MaxStable problem is *NP*-hard, we note that for many graphs the optimal solutions are obtained instantaneously. When the maximum stable set is not reachable in reasonable time for the four instances DSJC500.1, DSJC1000.1,  $le450_5a$  and  $le450_5b$ , we use an upper bound of  $\alpha(G)$ , computed by IncMaxCLQ within 2 h.

We conducted the experiment on the well-known COLOR [12] and DIMACS [6] benchmarks. The experimental results were obtained using an Intel Westmere Xeon E7-8837 (2.66 GHz) processor under Linux.

[Table 2](#) shows the experimental results. For each graph  $G$ , the columns indicate, in order, the name of the instance *Graph*, the number of vertices  $|V|$ , the density of the graph denoted by  $d$ , the best-known upper bound of the chromatic number  $k^*$  [7,17,26] (in parenthesis when  $k^*$  is not known to be optimal), the best-known upper bound of the chromatic sum  $\Sigma^*$  [16,17], the degree of the graph  $\Delta(G)$ , and the best known upper bound  $\alpha^*$  of the cardinality of a maximum stable set of  $G$  (in parenthesis when  $\alpha^*$  is not proved optimal).  $Time_S$  is the runtime in seconds to find  $UB_S$ , including the time for computing  $\alpha^*$ ,  $UB_{hmt}$  the result of Hajiabolhassan et al. given by [Property 4](#) using  $\Delta(G)$  and  $k^*$ ,  $UB_A$  our algebraic bound and  $UB_S$  our algorithmic bound based on  $\alpha^*$ .

According to [Table 2](#), the very simple algebraic bound  $UB_A$  is already better than  $UB_{hmt}$  for 43 instances out of 74, and  $UB_S$  obtains the best results for 70 instances out of 74. We do not observe any clear tendency correlated to any particular graph feature among the 70 graphs for which  $UB_S$  is better than  $UB_{hmt}$ . For example,  $UB_S$  is significantly better than  $UB_{hmt}$  for both large graphs, such as DSJC1000.5 with 1000 vertices, and small graphs, such as queen5\_5 with 25 vertices, and for both dense graphs, such as DSJC1000.9 with density 0.899, and sparse graphs, such as inithx.i.1 with density 0.05. In addition, the mean size of a color class of a graph (computed as  $|V|/k^*$  in the best known coloring) also is not a property to predict the quality of  $UB_S$ . For example,  $UB_S$  is better than  $UB_{hmt}$  for both graphs such as miles5000, in which the mean size of a color class is less than 2, and graphs such as flat1000-50-0, in which a color class contains 20 vertices in the average. Furthermore,  $UB_S$  can be better than  $UB_{hmt}$  even for those graphs for which a maximum stable set cannot be found in reasonable time. For example,  $UB_S$  is nearly two times smaller than  $UB_{hmt}$  for  $le450_5a$  and  $le450_5b$ , for which  $UB_S$  has to use an upper bound of  $\alpha(G)$  in [Algorithm 2](#) to derive a value.

Experimental results in [Table 2](#) show the effectiveness of our approach to derive a tight upper bound of the chromatic strength for general graphs. In particular,  $UB_S$  reaches and proves the optimality for eight graphs ( $le450_5c$ ,  $le450_5d$ , myciel3, queen5\_5, queen7\_7, queen8\_12, flat300-20-0 and flat1000-50-0), because  $k^* = \chi(G) = UB_S$  for these graphs.

#### 5. Conclusion

In this paper, we focused on the chromatic strength  $s(G)$  of a general graph  $G$ , defined as the minimum number of colors necessary to reach an optimal solution of the minimum sum coloring problem (*MSCP*). We proposed two new upper bounds of  $s(G)$ ,  $UB_A$  and  $UB_S$ , based on a valid coloring  $X$  of  $G$  and the notion of motifs. We first established a total order among all motifs related to  $G$ , and then derived  $UB_A$  and  $UB_S$  by identifying a motif worse than the valid coloring  $X$  but better than all subsequent motifs in the total order.

The derivation of  $UB_A$  does not use any structural information of  $G$ , apart from the valid coloring  $X$ , while the derivation of  $UB_S$  exploits the cardinality  $\alpha(G)$  (or an upper bound of  $\alpha(G)$ ) of a maximum stable set of  $G$ , based on the *major motif* notion introduced in this paper, in addition to  $X$ . We designed the algorithm  $CUB_S$  to compute  $UB_S$  in  $O(n^2)$  time.

We compared  $UB_A$  and  $UB_S$  with the upper bound  $UB_{hmt}$  of Hajiabolhassan et al. on the DIMACS and COLOR benchmarks. The experimental results show that  $UB_A$  is already better than  $UB_{hmt}$ , except for some graphs with low degree, and  $UB_S$  outperforms all others bounds in general, allowing a substantial reduction of the search space when solving the *MSCP*.

Our approach for deriving  $UB_S$  establishes a relation between  $s(G)$  and  $\alpha(G)$ . Although  $s(G)$  and  $\alpha(G)$  are both *NP*-hard to compute in theory, in practice  $\alpha(G)$  is considerably easier to compute than  $s(G)$ . In fact, the exact MaxClique algorithm IncMaxCLQ was able to compute  $\alpha(G)$  very quickly for most graphs in our experiment. We had to use an upper bound of  $\alpha(G)$  for only 4 graphs out of 74 when deriving  $UB_S$ . Using  $\alpha(G)$  computed by IncMaxCLQ and the relation established between  $s(G)$  and  $\alpha(G)$ , we were able to derive the exact value of  $s(G)$  for eight graphs for the first time in the literature.

Both  $UB_A$  and  $UB_S$  can be applied to the more general optimum cost chromatic partition problem. In the future, we plan to integrate more structural properties of  $G$  to further improve  $UB_S$ , and to develop efficient algorithms to solve the *MSCP* and the optimum cost chromatic partition (*OCCP*) problem.

#### Acknowledgment

This work is supported by the Ministry of Higher Education and Research of France.

**Table 2**

Comparison of our new algebraic bound ( $UB_A$ ) and algorithmic bound ( $UB_S$ ) with the bound of Hajiabolhassan et al. [8] ( $UB_{hmt}$ ).

Graph	$ V $	$d$	$k^*$	$\Sigma^*$	$\Delta(G)$	$\alpha^*$	Time <sub>s</sub>	$UB_{hmt}$	$UB_A$	$UB_S$
DSJC125.1	125	0.094	5	326	23	34	0	14	21	<b>11</b>
DSJC125.5	125	0.502	17	1012	75	10	0	46	43	<b>29</b>
DSJC125.9	125	0.898	44	2503	120	4	0	82	69	<b>58</b>
DSJC250.1	250	0.103	8	970	38	44	117	23	38	<b>21</b>
DSJC250.5	250	0.503	(28)	3210	147	12	0	88	77	<b>50</b>
DSJC250.9	250	0.896	72	8277	234	5	24	153	127	<b>105</b>
DSJC500.1	500	0.099	(12)	2836	68	(85)	1039	<b>40</b>	69	52
DSJC500.5	500	0.501	(48)	10886	286	13	16	167	145	<b>81</b>
DSJC500.9	500	0.901	(126)	29862	471	5	84	299	243	<b>185</b>
DSJC1000.1	1000	0.099	(20)	8991	127	(175)	1540	<b>74</b>	127	111
DSJC1000.5	1000	0.500	(83)	37575	551	15	408	317	271	<b>150</b>
DSJC1000.9	1000	0.899	(222)	103445	924	6	223	573	453	<b>347</b>
le450_5a	450	0.056	5	1350	42	(94)	2430	24	43	<b>13</b>
le450_5b	450	0.056	5	1350	42	(96)	1083	24	43	<b>15</b>
le450_5c	450	0.097	5	1350	66	90	2	36	43	<b>5</b>
le450_5d	450	0.096	5	1350	68	90	2	37	43	<b>5</b>
le450_15a	450	0.080	15	2632	99	75	45	57	67	<b>51</b>
le450_15b	450	0.080	15	2632	94	78	7	55	67	<b>52</b>
le450_15c	450	0.165	15	3487	139	41	1602	77	78	<b>50</b>
le450_15d	450	0.165	15	3505	138	41	2266	77	79	<b>50</b>
le450_25a	450	0.081	25	3153	128	91	0	77	74	<b>64</b>
le450_25b	450	0.081	25	3365	111	78	0	68	77	<b>66</b>
le450_25c	450	0.171	25	4515	179	47	24	102	91	<b>74</b>
le450_25d	450	0.172	25	4544	157	43	82	91	91	<b>72</b>
queen5_5	25	0.533	5	75	16	5	0	11	11	<b>5</b>
queen6_6	36	0.460	7	138	19	6	0	13	15	<b>10</b>
queen7_7	49	0.404	7	196	24	7	0	16	18	<b>7</b>
queen8_8	64	0.361	9	291	27	8	0	18	22	<b>10</b>
queen8_12	96	0.300	12	624	32	8	0	22	33	<b>12</b>
queen9_9	81	0.325	10	409	32	9	0	21	26	<b>11</b>
queen10_10	100	0.296	11	553	35	10	0	23	31	<b>12</b>
queen11_11	121	0.272	11	733	40	11	0	26	35	<b>14</b>
queen12_12	144	0.252	12	943	43	12	0	28	40	<b>15</b>
queen13_13	169	0.234	13	1191	48	13	0	31	46	<b>16</b>
queen14_14	196	0.219	14	1482	51	14	0	33	51	<b>18</b>
queen15_15	225	0.205	15	1814	56	15	0	36	57	<b>19</b>
queen16_16	256	0.193	16	2193	59	16	0	38	63	<b>21</b>
myciel3	11	0.363	4	21	5	5	0	5	5	<b>4</b>
myciel4	23	0.280	5	45	11	11	0	8	7	<b>6</b>
myciel5	47	0.218	6	93	23	23	0	15	10	<b>8</b>
myciel6	95	0.169	7	189	47	47	0	27	14	<b>11</b>
myciel7	191	0.130	8	381	95	95	0	52	20	<b>15</b>
fpsol2_i_1	496	0.094	65	3403	252	307	0	159	77	<b>75</b>
fpsol2_i_2	451	0.085	30	1668	346	261	0	188	50	<b>46</b>
fpsol2_i_3	425	0.096	30	1636	346	238	0	188	50	<b>46</b>
inithx.i.1	864	0.050	54	3676	502	566	0	278	75	<b>72</b>
inithx.i.2	645	0.067	31	2050	541	365	0	286	54	<b>48</b>
inithx.i.3	621	0.072	31	1986	542	360	0	287	53	<b>48</b>
mulsol.i.1	197	0.203	49	1957	121	100	0	85	60	<b>59</b>
mulsol.i.2	188	0.221	31	1191	156	90	0	94	45	<b>44</b>
mulsol.i.3	184	0.232	31	1187	157	86	0	94	45	<b>44</b>
mulsol.i.4	185	0.231	31	1189	158	86	0	95	45	<b>44</b>
mulsol.i.5	186	0.230	31	1160	159	88	0	95	45	<b>43</b>
school1	385	0.258	14	2674	282	41	2	148	68	<b>45</b>
school1-nsh	352	0.236	14	2392	232	39	0	123	64	<b>43</b>
zeroin.i.1	211	0.185	49	1822	111	120	0	80	57	<b>56</b>
zeroin.i.2	211	0.159	30	1004	140	127	0	85	40	<b>39</b>
zeroin.i.3	206	0.167	30	998	140	123	0	85	40	<b>39</b>
anna	138	0.052	11	276	71	80	0	41	17	<b>14</b>
david	87	0.108	11	237	82	36	0	47	18	<b>15</b>
huck	74	0.111	11	243	53	27	0	32	19	<b>16</b>
jean	80	0.080	10	217	36	38	0	23	17	<b>15</b>
games120	120	0.089	9	443	13	22	0	<b>11</b>	26	15
flat300-20-0	300	0.476	20	3150	160	15	0	90	76	<b>20</b>
flat300-26-0	300	0.482	26	3966	158	12	0	92	86	<b>36</b>
flat300-28-0	300	0.483	28	4238	162	12	0	95	89	<b>49</b>

(continued on next page)

Table 2 (continued)

Graph	$ V $	$d$	$k^*$	$\Sigma^*$	$\Delta(G)$	$\alpha^*$	Time <sub>s</sub>	$UB_{hmt}$	$UB_A$	$UB_S$
flat1000-50-0	1000	0.490	50	25500	520	20	263	285	222	<b>50</b>
flat1000-60-0	1000	0.492	60	30100	524	17	296	292	242	<b>77</b>
flat1000-76-0	1000	0.493	76	37164	532	15	466	304	269	<b>145</b>
miles250	128	0.047	8	325	16	44	0	<b>12</b>	20	14
miles500	128	0.143	20	705	38	18	0	29	34	<b>25</b>
miles750	128	0.259	31	1173	64	12	0	48	46	<b>38</b>
miles1000	128	0.395	42	1666	86	8	0	64	56	<b>47</b>
miles1500	128	0.639	73	3354	106	5	0	90	81	<b>77</b>

## References

- [1] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, Tami Tamir, On chromatic sums and distributed resource allocation, *Inform. and Comput.* 140 (2) (1998) 183–202.
- [2] Amotz Bar-Noy, Guy Kortsarz, Minimum color sum of bipartite graphs, *J. Algorithms* 28 (2) (1998) 339–365.
- [3] Una Benlic, Jin-Kao Hao, A study of breakout local search for the minimum sum coloring problem, in: *Simulated Evolution and Learning*, Springer, 2012, pp. 128–137.
- [4] Flavia Bonomo, Mario Valencia-Pabon, Minimum sum coloring of  $P_4$ -sparse graphs, *Electron. Notes Discrete Math.* 35 (2009) 293–298.
- [5] Flavia Bonomo, Mario Valencia-Pabon, On the minimum sum coloring of  $P_4$ -sparse graphs, *Graphs Combin.* 30 (2) (2014) 303–314.
- [6] <http://www.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/>.
- [7] Stefano Gualandi, Federico Malucelli, Exact solution of graph coloring problems via constraint programming and column generation, *INFORMS J. Comput.* 24 (1) (2012) 81–100.
- [8] Hossein Hajiabolhassan, Mojtaba L. Mehrabadi, Ruzbeh Tusserkani, Minimal coloring and strength of graphs, *Discrete Math.* 215 (1) (2000) 265–270.
- [9] Magnús M. Halldórsson, Guy Kortsarz, Multicoloring: Problems and techniques, in: *International Symposium on Mathematical Foundations of Computer Science*, Springer, 2004, pp. 25–41.
- [10] Magnus M. Halldórsson, Guy Kortsarz, Hadas Shachnai, Minimizing average completion of dedicated tasks and interval graphs, in: Michel Goemans, Klaus Jansen, José D.P. Rolim, Luca Trevisan (Eds.), *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques: 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2001 and 5th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2001 Berkeley, CA, USA, August 18–20, 2001 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 114–126.
- [11] Godfrey H. Hardy, Srinivasa Ramanujan, Asymptotic formulæ in combinatory analysis, *Proc. Lond. Math. Soc.* 2 (1) (1918) 75–115.
- [12] <http://mat.gsia.cmu.edu/COLOR/instances.html>.
- [13] Klaus Jansen, The optimum cost chromatic partition problem, in: *Algorithms and Complexity*, Springer, 1997, pp. 25–36.
- [14] Klaus Jansen, Approximation results for the optimum cost chromatic partition problem, *J. Algorithms* 34 (1) (2000) 54–89.
- [15] Yan Jin, Jean-Philippe Hamiez, Jin-Kao Hao, Algorithms for the minimum sum coloring problem: a review, *Artif. Intell. Rev.* (2015) 1–28.
- [16] Yan Jin, Jin-Kao Hao, Hybrid evolutionary search for the minimum sum coloring problem of graphs, *Inform. Sci.* 352 (2016) 15–34.
- [17] Yan Jin, Jin-Kao Hao, Jean-Philippe Hamiez, A memetic algorithm for the minimum sum coloring problem, *Comput. Oper. Res.* 43 (2014) 318–327.
- [18] Richard M. Karp, Reducibility among combinatorial problems, in: *Complexity of Computer Computations*, Springer, 1972, pp. 85–103.
- [19] Leo G. Kroon, Arunabha Sen, Haiyong Deng, Asim Roy, The optimal cost chromatic partition problem for trees and interval graphs, in: *International Workshop on Graph-Theoretic Concepts in Computer Science*, Springer, 1996, pp. 279–292.
- [20] Ewa Kubicka, Allen J. Schwenk, An introduction to chromatic sums, in: *Proceedings of the 17th Conference on ACM Annual Computer Science Conference*, ACM, 1989, pp. 39–45.
- [21] Clément Lecat, Chu-Min Li, Corinne Lucet, Yu Li, Exact methods for the minimum sum coloring problem, in: *Doctoral Program of Constraint Programming, CP-DP'15*, 2015, pp. 61–69.
- [22] Clément Lecat, Corinne Lucet, Chu-Min Li, New lower bound for the minimum sum coloring problem, in: *AAAI Conference on Artificial Intelligence*, AAAI, 2017.
- [23] Chu-Min Li, Zhiwen Fang, Ke Xu, Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem, in: *2013 IEEE 25th International Conference on Tools with Artificial Intelligence, ICTAI, IEEE*, 2013, pp. 939–946.
- [24] Yu Li, Corinne Lucet, Aziz Moukrim, Kaoutar Sghiouer, Greedy algorithms for the minimum sum coloring problem, in: *Logistique Et Transports*, 2009, pp. LT-027.
- [25] Michal Malafiejski, in: Marek Kubale (Ed.), *Sum Coloring of Graphs*, in: *Graph Coloring*, 2004, pp. 55–66.
- [26] Enrico Malaguti, Michele Monaci, Paolo Toth, An exact approach for the vertex coloring problem, *Discrete Optim.* 8 (2) (2011) 174–190.
- [27] John Mitchem, Patrick Morriss, On the cost-chromatic number of graphs, *Discrete Math.* 171 (1) (1997) 201–211.
- [28] Aziz Moukrim, Kaoutar Sghiouer, Corinne Lucet, Yu Li, Lower bounds for the minimal sum coloring problem, *Electron. Notes Discrete Math.* 36 (2010) 663–670.
- [29] Sara Nicoloso, Sum coloring and interval graphs: a tight upper bound for the minimum number of colors, *Discrete Math.* 280 (1) (2004) 251–257.
- [30] Mohammad R. Salavatipour, On sum coloring of graphs, *Discrete Appl. Math.* 127 (3) (2003) 477–488.
- [31] Arunabha Sen, Haiyong Deng, Sumanta Guha, On a graph partition problem with application to VLSI layout, *Inform. Process. Lett.* 43 (2) (1992) 87–94.
- [32] Kaoutar Sghiouer, Li Yu, Corinne Lucet, Aziz Moukrim, A memetic algorithm for the minimum sum coloring problem, in: *Confrence Internationale en Recherche Opérationnelle, CIRO'2010*, 2010.
- [33] Carsten Thomassen, Paul Erdős, Yousef Alavi, Paresh J. Malde, Allen J. Schwenk, Tight bounds on the chromatic sum of a connected graph, *J. Graph Theory* 13 (3) (1989) 353–357.

## New Lower Bound for the Minimum Sum Coloring Problem

Clément Lecat, Corinne Lucet, Chu-Min Li

Laboratoire de Modélisation, Information et Système EA 4290,  
 Université de Picardie Jules Verne, Amiens, France  
 {clement.lecat, corinne.lucet, chu-min.li}@u-picardie.fr

### Abstract

The Minimum Sum Coloring Problem (MSCP) is an NP-Hard problem derived from the graph coloring problem (GCP) and has practical applications in different domains such as VLSI design, distributed resource allocation, and scheduling. There exist few exact solutions for MSCP, probably due to its search space much more elusive than that of GCP. On the contrary, much effort is spent in the literature to develop upper and lower bounds for MSCP. In this paper, we borrow a notion called motif, that was used in a recent work for upper bounding the minimum number of colors in an optimal solution of MSCP, to develop a new algebraic lower bound called  $LBM\Sigma$  for MSCP. Experiments on standard benchmarks for MSCP and GCP show that  $LBM\Sigma$  is substantially better than the existing lower bounds for several families of graphs.

### Introduction

The Minimum Sum Coloring Problem (MSCP) is a NP-hard problem introduced in 1989 by Kubicka and Schwenk (Kubicka and Schwenk 1989). As the widely studied Graph Coloring Problem (GCP), MSCP also consists in assigning one color to each vertex of a given graph  $G$ , respecting the neighborhood constraints. But, MSCP considers in addition a weight associated with each color. While an optimal solution of GCP is a coloring of  $G$  that uses the minimum number of colors, called the chromatic number of  $G$  and denoted by  $\chi(G)$ , an optimal solution of MSCP is a coloring of  $G$  such that the sum of weights associated with the used colors is minimum. This minimum sum is called the *chromatic sum* of  $G$  and is denoted by  $\Sigma(G)$ . The minimum number of colors required in an optimal solution of MSCP is called the *strength* (or *chromatic strength*) of  $G$ , and is denoted by  $s(G)$ . It holds that  $\chi(G) \leq s(G)$ .

Apart from the theoretical interest, MSCP has practical applications in different domains such as VLSI design, distributed resource allocation, scheduling (Bar-Noy et al. 1998; Malafiejski 2004), and more particularly in the field of the management of networks. Indeed, the quality of service (QoS) through a network of customers can easily be reduced to an instance of MSCP of the corresponding graph. To illustrate this, we consider an Allocation

Resource Problem (ARP) (Chandy and Misra 1984; Bar-Noy et al. 1998) with a set of jobs  $\{J_1, J_2, \dots, J_n\}$  that have to run on some computer servers  $\{P_1, P_2, \dots, P_m\}$ . Each job needs a unit of time to be completed and two jobs requiring the same server cannot be executed simultaneously. The set of constraints can be represented by a conflict graph, where a vertex represents a job and an edge represents a conflict between two jobs. The aim is to compute a job scheduling that minimizes the average response time for jobs. Such a solution is given by an optimal sum coloring of the conflict graph where colors are time units. Figure 1 shows an ARP instance of 8 jobs and 7 computer servers. Figure 2 shows the associated conflict graph, from which GCP resolution provides the following scheduling :  $\{J_1, J_2, J_3, J_5\}$  at time 1 and  $\{J_4, J_6, J_7, J_8\}$  at time 2 with the average response time  $(4 \times 1 + 4 \times 2)/8 = 1.5$ . MSCP resolution provides the following scheduling :  $\{J_1, J_2, J_3, J_6, J_7, J_8\}$  at time 1,  $\{J_4\}$  at time 2 and  $\{J_5\}$  at time 3 with the average response time  $(6 \times 1 + 2 + 3)/8 = 1.3$ . So, even if only 2 units of time are sufficient to schedule the 8 jobs, it is better to use a third unit of time to reduce the average response time and to increase the global QoS.

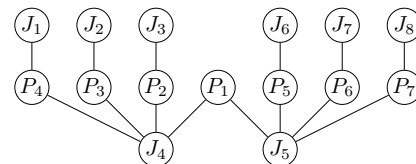


Figure 1: An allocation resource problem

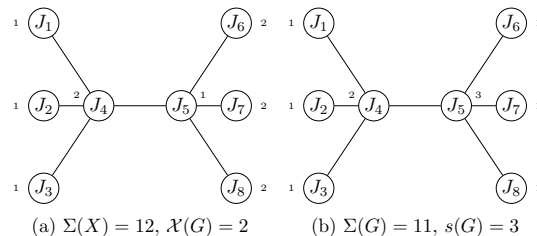


Figure 2: GCP (a) and MSCP (b) on the conflict graph



Recently, there is a growing interest on MSCP because of its practical applications. Unfortunately, solving MSCP is generally much harder than solving GCP, because it is substantially harder to reduce the number of colors to be considered when solving MSCP than when solving GCP. In fact, when a GCP algorithm finds a valid coloring solution with  $k$  colors, it immediately prunes the sub-space with  $k$  or more colors. However, a MSCP algorithm cannot do this, because an optimal solution of MSCP can involve more than  $k$  colors. For this reason, most MSCP algorithms in the literature are heuristic/meta-heuristic (Li et al. 2009; Sghiouer et al. 2010; Jin, Hao, and Hamiez 2014; Benlic and Hao 2012; Moukrim et al. 2010), providing approximate solutions to the problem. The few exact MSCP algorithms are based on the branch-and-bound schema (Lecat et al. 2015), linear programming (Wang, Hao, and Glover 2013), CSP solving (Lecat et al. 2015; Minot, Ndiaye, and Solnon 2016), or MaxSAT/MinSAT solving (Lecat et al. 2015), and can only solve small graphs.

In this paper, we borrow a notion called *motif*, that was used in (Lecat, Lucet, and Li 2016) for upper bounding the *chromatic strength*  $s(G)$  of a graph  $G$ , to develop a new algebraic lower bound called  $LBM\Sigma$  for  $\Sigma(G)$ . A tight lower bound  $lb$  of  $\Sigma(G)$  is useful, as illustrated by the following two examples.

- A heuristic MSCP algorithm generally gives a valid coloring  $X$  of  $G$  and the corresponding sum  $\Sigma(X)$  of weights of the used colors. The quality of this solution can be estimated using  $lb$ . In particular, if  $lb = \Sigma(X)$ ,  $\Sigma(X)$  is proved to be an optimal solution of MSCP.
- A branch-and-bound MSCP algorithm needs to compute  $lb$  at every search tree node. Let  $X_{best}$  be the best valid coloring found so far. If  $lb \geq \Sigma(X_{best})$ , then search below the tree node can be pruned.

This paper is organized as follows. Section 2 presents the necessary definitions. Section 3 presents the notion of motifs and summarizes properties introduced in (Lecat, Lucet, and Li 2016). Section 4 describes our approach based on motifs to compute the lower bound  $LBM\Sigma$  of the chromatic sum  $\Sigma(G)$ . Section 5 analyses the empirical results. Finally, Section 6 concludes the paper.

## Basic Definitions

Let  $G = (V, E)$  be an undirected graph, where  $V$  is the set of  $n$  vertices ( $|V| = n$ ) and  $E \subseteq V^2$  is the set of edges. A coloring of  $G$  is a function  $c : V \mapsto \{1, 2, \dots, k\}$  that assigns a color  $c(v)$ , represented by an integer, to each vertex  $v \in V$ . A coloring is valid iff  $c(v) \neq c(v') \forall (v, v') \in E$ . The color class  $X_i$  ( $1 \leq i \leq k$ ), is the set of vertices  $v$  such that  $c(v) = i$ . A valid coloring is denoted by  $X = \{X_1, X_2, \dots, X_k\}$ . For MSCP, a weight  $w_i$  is associated with each color  $i$ . In this paper, we consider that  $w_i = i$ . We denote by  $\Sigma(X)$  the sum of color weights of the valid coloring  $X$ , calculated by Equation 1.

$$\Sigma(X) = 1 \times |X_1| + 2 \times |X_2| + \dots + k \times |X_k| \quad (1)$$

An optimal solution of MSCP for a graph  $G$ , denoted by  $\Sigma(G)$ , is defined as follows :

$$\Sigma(G) = \min\{\Sigma(X) | X \text{ is a valid coloring of } G\} \quad (2)$$

Referring to the graph in Figure 3,  $X = \{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is a valid coloring where the vertices  $c, d$  and  $f$  are colored with the color 1,  $a$  and  $e$  with the color 2 and  $b$  with the color 3.  $X$  is an optimal solution for MSCP, and the chromatic sum is  $\Sigma(G) = \Sigma(X) = 10$ .

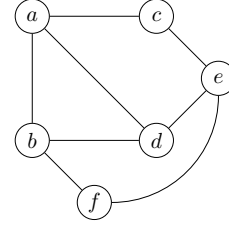


Figure 3: A simple graph

A stable set of a graph  $G = (V, E)$  is a subset  $S \subseteq V$  such that  $\forall (v, v') \in S^2, (v, v') \notin E$ . A clique of  $G$  is a subset  $C \subseteq V$  such that  $\forall (v, v') \in C^2, (v, v') \in E$ . We denote by  $\alpha(G)$  the cardinality of a maximum stable set of graph  $G$ , called the *independence number* of  $G$ .

$$\alpha(G) = \max\{|S| \mid \forall (v, v') \in S^2, (v, v') \notin E\} \quad (3)$$

A graph  $\overline{G} = (V, \overline{E})$ , where  $\overline{E} = \{(v, v') \in V^2 \mid (v, v') \notin E\}$ , is called the *complement graph* of  $G$ . A clique of  $G$  is clearly a stable set of  $\overline{G}$ , and vice versa. Since the vertices in a clique of  $G$  need different colors to be colored,  $\alpha(\overline{G})$  is a lower bound of the chromatic number of  $G$ , which itself is a lower bound of the strength of  $G$  :

$$\alpha(\overline{G}) \leq \chi(G) \leq s(G) \quad (4)$$

The permutation of color classes of a valid coloring  $X$  has no impact on its validity. Thus, the permutation set of color classes of  $X$  forms an equivalent class, denoted by  $\Psi(X)$ . The colorings of  $\Psi(X)$  are symmetric, and use the same number of colors. However, the sum of color weights of the symmetric colorings is not necessarily the same. We can notice it on Figure 3, where the sum associated to  $\{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is 10, while the sum associated to the symmetric coloring  $\{\{a, e\}_1, \{b\}_2, \{c, d, f\}_3\}$  is 13. Because MSCP consists in minimizing the sum of weights, it could be interesting to focus on colorings having the smallest sum among those in  $\Psi(X)$ . We call such a coloring a *major coloring*, that is defined as follows :

**Definition 1** A major coloring, denoted by  $X^m$ , is a coloring  $X^m = \{X_1, X_2, \dots, X_k\}$  such that  $|X_1| \geq |X_2| \geq \dots \geq |X_k|$ .

**Property 1** Let  $\Psi(X)$  be the set of symmetric colorings of  $X$ , and  $X^m$  be a major coloring of  $\Psi(X)$ , then  $\forall X' \in \Psi(X), \Sigma(X^m) \leq \Sigma(X')$ .

Property 1 is a consequence of the specific structure of major colorings that can be summarized by the intuitive idea that the smallest weight should be assigned to the vertices in the largest color class. Referring to Figure 3, the coloring  $\{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is a major coloring with sum equal to 10. Consequently, we only consider major colorings in this paper, which allow to define the notion of *motifs* in the next section. Such motifs are used to construct our lower bound of the chromatic sum  $LBM\Sigma$ .

### Motifs

The notion of motifs was used in (Bonomo and Valencia-Pabon 2009; 2014) to solve MSCP for P4-sparse graphs and in (Lecat, Lucet, and Li 2016) to define an algebraic upper bound and an algorithmic upper bound for the chromatic strength of the graph. In this section, we briefly recall this notion and its properties useful in the construction of  $LBM\Sigma$ .

A motif is a representation of a major coloring by a non-increasing sequence of integers as described in Definition 2.

**Definition 2** *The motif associated with a major coloring  $X = \{X_1, X_2, \dots, X_k\}$  is a non-increasing sequence of positive integers, denoted by  $p = (|X_1|, |X_2|, \dots, |X_k|)$ . The  $i^{th}$  integer is  $p[i] = |X_i|$ . The sum of color weights corresponding to  $p$  can be computed using Equation 5. The length of  $p$  is the number of colors used by  $X$ , denoted by  $|p|$  ( $|p| = k$ ).*

$$\Sigma(X) = \Sigma(p) = 1 \times p[1] + 2 \times p[2] + \dots + k \times p[k] \quad (5)$$

According to this definition, the motif associated with  $X = \{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  is  $p = (3, 2, 1)$ , where  $p[1] = 3$ ,  $p[2] = 2$  and  $p[3] = 1$ .  $\Sigma(X) = \Sigma(p) = 10$ . We note that different major colorings can share the same associated motif. Indeed, the colorings  $X^a = \{\{c, d, f\}_1, \{a, e\}_2, \{b\}_3\}$  and  $X^b = \{\{c, d, f\}_1, \{b, e\}_2, \{a\}_3\}$  both correspond to  $p = (3, 2, 1)$ , their sum of color weights are equal to  $\Sigma(p)$ . Thus, if we consider motifs instead of major colorings, we reduce further the representation of the search space.

For a graph  $G = (V, E)$  with  $|V| = n$ , we denote by  $\phi(n)$  the set of all possible motifs associated with major colorings of  $G$ , and  $\phi(n, k)$  the set of motifs such that  $|p| = k$ . The sets  $\phi(n, k)$  in  $\phi(n)$  are sorted by  $k$  in increasing order and the motifs in  $\phi(n, k)$  are sorted in the decreasing lexicographical order defined below.

**Definition 3** *Let  $p_k$  and  $q_k$  be two motifs in  $\phi(n, k)$ . We say that  $p_k$  lexicographically precedes (or is greater than)  $q_k$  iff  $p_k[1] > q_k[1]$  or  $\exists t > 0$  such that  $p_k[x] = q_k[x]$  for each  $x < t$  and  $p_k[t] > q_k[t]$ .*

The decreasing lexicographical order allows to assign an index  $i$  to each motif in  $\phi(n, k)$ :  $p_k^i$  is the  $i^{th}$  motif in  $\phi(n, k)$  in the decreasing lexicographical order. Table 1 lists the motifs of  $\phi(8)$  and the corresponding sums of color weights.

We observe that the motif  $p_3^1 = (6, 1, 1)$  is lexicographically greater than the motif  $p_3^3(4, 2, 2)$ , so  $p_3^1$  precedes  $p_3^3$  in  $\phi(8, 3)$ . All motifs in  $\phi(8, 3)$  precede the motifs in  $\phi(8, 4)$ . This order is of great importance as shown in (Lecat, Lucet,

and Li 2016) to compute the upper bound of the chromatic strength  $s(G)$ . We take the same way to construct the lower bound of MSCP,  $LBM\Sigma$ , based on the dominance relation between motifs defined below.

$\phi(n,k)$	$p \in \phi(n,k)$	$\Sigma(p)$
$\phi(8,1)$	(8)	8
$\phi(8,2)$	(7,1)	9
	(6,2)	10
	(5,3)	11
	(4,4)	12
$\phi(8,3)$	(6,1,1)	11
	(5,2,1)	12
	(4,3,1)	13
	(4,2,2)	14
	(3,3,2)	15
$\phi(8,4)$	(5,1,1,1)	14
	(4,2,1,1)	15
	(3,3,1,1)	16
	(3,2,2,1)	17
	(2,2,2,2)	20
$\phi(8,5)$	(4,1,1,1,1)	18
	(3,2,1,1,1)	19
	(2,2,2,1,1)	21
$\phi(8,6)$	(3,1,1,1,1,1)	23
	(2,2,1,1,1,1)	24
$\phi(8,7)$	(2,1,1,1,1,1,1)	29
$\phi(8,8)$	(1,1,1,1,1,1,1,1)	36

Table 1:  $\phi(8)$ : the set of all possible motifs for  $n = 8$  vertices

**Definition 4** *Let  $p$  and  $q$  be two motifs in  $\phi(n)$ . We say that  $p$  dominates  $q$ , denoted by  $p \succeq q$ , if and only if  $\forall t$  such that  $1 \leq t \leq \min\{|p|, |q|\}$ ,  $\sum_{x=1}^t p[x] \geq \sum_{x=1}^t q[x]$ .*

**Example 1** *Let  $p$  and  $q$  be two motifs of a graph  $G$  with 8 vertices,  $p = (5, 2, 1)$  and  $q = (4, 3, 1)$ .*

*For  $t = 1 : 5 > 4 ;$*

*For  $t = 2 : 5 + 2 \geq 4 + 3 ;$*

*For  $t = 3 : 5 + 2 + 1 \geq 4 + 3 + 1.$*

*Thus  $p \succeq q$ .*

**Property 2** *Let  $G$  be a graph,  $p$  and  $q$  two motifs, respectively associated with two valid colorings  $X$  and  $X'$  of  $G$ . If  $p \succeq q$ , then  $\Sigma(X) \leq \Sigma(X')$ .*

We note that the dominance relation is a partial order, contrary to the decreasing lexicographical order which is total. Referring to Table 1,  $(4, 3, 1) \succeq (3, 2, 1, 1, 1)$ , but  $(3, 1, 1, 1, 1)$  and  $(2, 2, 2, 2)$  are incomparable, because  $(3, 1, 1, 1, 1) \not\succeq (2, 2, 2, 2)$  and  $(2, 2, 2, 2) \not\succeq (3, 1, 1, 1, 1)$ .

The first motif in  $\phi(n, k)$  in the decreasing lexicographical order is  $p_k^1$ . It has the following specific structure:

$$(n - k + 1, \overbrace{1, 1, \dots, 1}^{k-1 \text{ times}})$$



In  $\phi(8)$ ,  $p_2^1 = (7, 1)$ ,  $p_3^1 = (6, 1, 1)$ ,  $p_4^1 = (5, 1, 1, 1)$ , etc. The Properties 3 and 4 are consequences of this structural property.

**Property 3** Let  $G(V, E)$  be a graph, and  $k \leq n$  an integer. The motif  $p_k^1 = (n - k + 1, 1, 1, \dots, 1)$  dominates all other motifs in  $\phi(n, k)$ .

**Proof 1** Let  $q$  be a motif in  $\phi(n, k)$ . Then  $\forall t$  such that  $1 \leq t \leq k$ ,  $\sum_{x=t+1}^k p_k^1[x] \leq \sum_{x=t+1}^k q[x]$ , because  $p_k^1[x] = 1 \leq q[x]$  when  $x > 1$ . So,  $\sum_{x=1}^t p_k^1[x] = n - \sum_{x=t+1}^k p_k^1[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ .

**Property 4** Let  $G(V, E)$  be a graph, and  $k$  and  $k'$  be two integers such that  $1 \leq k < k' \leq n$ .  $p_k^1 = (n - k + 1, 1, 1, \dots, 1)$  dominates  $p_{k'}^1 = (n - k' + 1, 1, 1, \dots, 1)$ .

**Proof 2** According to the specific structure of a first motif,  $\forall t \leq k$ , we have  $\sum_{x=1}^t p_k^1[x] = n - k + 1 + t - 1 \geq n - k' + 1 + t - 1 = \sum_{x=1}^t p_{k'}^1[x]$ .

Consequently, we conclude that  $p_k^1$  dominates all its successors in  $\phi(n)$  in the decreasing lexicographical order, and then, its associated sum of color weights is the smallest one over all the colorings that use at least  $k$  colors, in the search space.

### Algebraic Lower Bound for Chromatic Sum

An algebraic lower bound  $LB_{Kok}$  for the chromatic sum  $\Sigma(G)$  was proposed in (Kokosiński and Kwarciány 2007). In this Section, we firstly re-state  $LB_{Kok}$  in terms of motifs, and then present the new lower bound  $LBM\Sigma$  of  $\Sigma(G)$ .

As mentioned in the previous section, Properties 3 and 4 mean that the motif  $p_k^1$  dominates all the motifs  $p_{k'}^i$  such that  $i \geq 1$  and  $k' \geq k$ . That is to say, we cannot find an MSCP solution  $X$  that uses  $k$  or more colors, such that  $\Sigma(X) < \Sigma(p_k^1)$ . More formally:

$$\forall q \in \bigcup_{x=k}^n \{\phi(n, x)\}, p_k^1 \succeq q \text{ and } \Sigma(p_k^1) \leq \Sigma(q).$$

The sum of color weights associated with  $p_k^1$  is easily computed using Equation 6. Moreover, we know that the minimum number of colors to color a given graph  $G$  is its chromatic number  $\chi(G)$ . Then  $\Sigma(p_{\chi(G)}^1)$  is the lowest sum of color weights that can be reached. We deduce the lower bound  $LB_{Kok}$  given by Equation 7, equal to  $\Sigma(p_{\chi(G)}^1)$ . Of course, the chromatic number is not necessarily known for  $G$ . In that case, any lower bound of  $\chi(G)$  (e.g., the cardinality of a maximum clique of  $G$ ) can replace  $\chi(G)$  in Equation 7 to give a lower bound of  $\Sigma(G)$ .

$$\Sigma(p_k^1) = (n - k) + \frac{k \times (k + 1)}{2} \quad (6)$$

$$LB_{Kok} = (n - \chi(G)) + \frac{\chi(G) \times (\chi(G) + 1)}{2} \quad (7)$$

The main drawback of  $LB_{Kok}$  is the lack of consideration of the structural properties of the graph  $G$ . Hence, we propose to consider one of the characteristics of  $G$ , the independence number  $\alpha(G)$  combined with the motif representation to improve the lower bound. Indeed, since each integer in a motif represents the cardinality of one color class and each color class is a stable set, any motif containing an integer greater than  $\alpha(G)$  cannot match a valid coloring of  $G$ . In particular, if  $p_k^i[1] > \alpha(G)$ ,  $p_k^i$  can be excluded from the search space.

The principal idea of the construction of  $LBM\Sigma$  consists in finding a motif  $p_k^i$  ( $i \geq 1$  and  $1 \leq k \leq n$ ) such that  $p_k^i[1] = \alpha(G)$  and  $p_k^i$  dominates all its successors  $q$  in  $\phi(n)$  with  $q[1] \leq \alpha(G)$ . If such a motif exists, the corresponding associated sum  $\Sigma(p_k^i)$  is a lower bound for  $\Sigma(G)$ . For this purpose, we borrow the notion of *major motif* that has been introduced in (Lecat, Lucet, and Li 2016) based on the following observation : Let  $X$  be a valid major coloring, it could be possible to find two color classes  $X_i$  and  $X_j$ , with  $|X_i| \geq |X_j|$  and  $i < j$ , such that shifting a vertex of  $X_j$  to  $X_i$  induces a valid coloring  $X'$  with  $\Sigma(X') < \Sigma(X)$ . Thus, we define for a motif  $p$  the *left-shifting operation* as follows.

**Definition 5** Let  $p$  be a motif. The left-shifting operation is the transformation of  $p$  to  $p'$  such that  $p'[i] = p[i] + 1$ ,  $p'[j] = p[j] - 1$  for two positive integers  $i < j$ , and  $p'[x] = p[x]$  for  $x \neq i, x \neq j$ , and  $p'$  remains a non-increasing sequence of positive integers.

Because  $p'$  should remain a non-increasing sequence of positive integers, the left-shifting operation is not always applicable. Thus, a motif in  $\phi(n, k)$  that cannot be transformed into another motif in  $\phi(n, k)$  by a left-shifting operation without incrementing the first integer is called *major motif* in (Lecat, Lucet, and Li 2016). A left-shifting operation incrementing the first integer of a motif is excluded in the notion of major motif, because incrementing the first integer may make it bigger than  $\alpha(G)$ , so that the obtained motif does not correspond to any valid coloring of  $G$ .

Roughly speaking, a motif is major if it gives the smallest sum of color weights among all motifs that use the same number of colors and in which the first integer is the same. Referring to Table 1, there are two motifs in  $\Phi(8, 3)$  in which the first integer is 4: (4, 3, 1) and (4, 2, 2), the motif (4, 3, 1) is major; and there are two motifs in  $\Phi(8, 4)$  in which the first integer is 3: (3, 3, 1, 1) and (3, 2, 2, 1), the motif (3, 3, 1, 1) is major. Intuitively, a major motif in  $\phi(n, k)$  contains the maximum number ( $\beta$ ) of integers equal to its first integer, denoted by  $\lambda$  ( $\lceil \frac{n}{k} \rceil \leq \lambda \leq n - k + 1$ ). The remaining value of  $n$  (i.e.  $n - \beta \times \lambda$ ) should be partitioned into  $k - \beta$  positive integers. So  $\beta$  is the maximum integer satisfying  $n - \beta \times \lambda \geq k - \beta$ , or  $\beta \leq \frac{n-k}{\lambda-1}$  after excluding the trivial motif (1, 1, ..., 1) and assuming  $\lambda > 1$ . So,  $\beta = \lfloor \frac{n-k}{\lambda-1} \rfloor$ .

A major motif is formally defined in (Lecat, Lucet, and Li 2016) as follows.

**Definition 6** Let  $\lambda$  and  $\beta$  be two integers such that  $\lceil \frac{n}{k} \rceil \leq$

$\lambda \leq n - k + 1$  and  $\beta = \left\lfloor \frac{n-k}{\lambda-1} \right\rfloor$ . A motif  $p_k^i$  in  $\phi(n, k)$  is major if:

1.  $p_k^i[x] = \lambda$ , if  $1 \leq x \leq \beta$ ;
2.  $p_k^i[x] = n - \beta \times \lambda - (k - \beta - 1)$ , if  $x = \beta + 1$ ;
3.  $p_k^i[x] = 1$ , if  $\beta + 1 < x \leq k$ .

It is easy to see that the left-shifting operation is not applicable to a major motif without incrementing its first integer. The following property gives an insight to the notion of motifs.

**Property 5** (1) Let  $p$  be a motif in  $\phi(n, k)$ , then  $\left\lceil \frac{n}{k} \right\rceil \leq p[1] \leq n - k + 1$ . (2) Let  $O$  denote the decreasing lexicographical order. For any number  $\lambda$  such that  $\left\lceil \frac{n}{k} \right\rceil \leq \lambda \leq n - k + 1$ , consider the subset of motifs  $\{p \mid p[1] = \lambda\}$  of  $\phi(n, k)$ , the left-shifting operation is not applicable to the greatest motif  $m$  w.r.t.  $O$  in this subset without incrementing  $m[1]$ ; furthermore, if a motif  $q$  in this subset is not the greatest motif w.r.t.  $O$ , the left-shifting operation is always applicable to  $q$  to transform it into a greater one w.r.t.  $O$ .

**Proof 3** (1) If  $p[1] > n - k + 1$ , then  $p[1] + p[2] + \dots + p[k] > n$ , because  $p[x] \geq 1$  for  $x > 1$ . If  $p[1] < \left\lceil \frac{n}{k} \right\rceil$ , then  $p[1] + p[2] + \dots + p[k] \leq k \times \left\lceil \frac{n}{k} \right\rceil - k < n$ .

(2) It is easy to see that a left-shifting operation applied to a motif  $p$  results in a motif  $p'$  such that  $p'$  precedes  $p$  w.r.t.  $O$ . So, the left-shifting operation cannot be applied to the greatest motif  $m$  w.r.t.  $O$  in  $\{p \mid p[1] = \lambda\}$  of  $\phi(n, k)$  without incrementing  $m[1]$ . Let  $q$  be a motif in the subset such that  $m$  precedes  $q$  w.r.t.  $O$ . By definition, there is a number  $t > 0$  such that  $m[x] = q[x]$  for  $x < t$  and  $m[t] > q[t]$ . Let  $y$  be the greatest number such that  $m[y] < q[y]$  ( $y$  exists because  $\sum_{x=1}^k q[x] = \sum_{x=1}^k m[x] = n$ ). We can apply the left-shifting operation by decrementing  $q[y]$  and incrementing  $q[t]$  to obtain a motif preceding  $q$  w.r.t.  $O$ .

Property 5 says that only the greatest motif w.r.t.  $O$  in the subset  $\{p \mid p[1] = \lambda\}$  of  $\phi(n, k)$  is major, meaning that, although the total number of motifs in  $\phi(n)$  is exponential, the number of major motifs is only quadratic.

The following properties are the consequences of the structure of major motifs.

**Property 6** Let  $p$  and  $q$  be two motifs in  $\phi(n, k)$  such that  $p$  is major and  $p[1] = q[1]$ , then  $p \succeq q$ .

**Proof 4** Let  $\beta = \left\lfloor \frac{n-k}{p[1]-1} \right\rfloor$ . Since  $p$  is major, we have  $p[1] = p[2] = \dots = p[\beta] = q[1] \geq q[2] \geq \dots \geq q[\beta]$ . So,  $\forall t$  such that  $1 \leq t \leq \beta$ ,  $\sum_{x=1}^t p[x] \geq \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta + 1 \leq t < k$ , we have  $p[t+1] = p[t+2] = \dots = p[k] = 1 \leq q[k] \leq q[k-1] \leq \dots \leq q[t+1]$ .

So,  $\sum_{x=1}^t p[x] = n - \sum_{x=t+1}^k p[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ .

Therefore,  $p \succeq q$ .

**Property 7** Let  $p$  and  $q$  be two major motifs in  $\phi(n, k)$ . If  $p[1] > q[1]$  then  $p \succeq q$ .

**Proof 5** Let  $\beta_p = \left\lfloor \frac{n-k}{p[1]-1} \right\rfloor$  and  $\beta_q = \left\lfloor \frac{n-k}{q[1]-1} \right\rfloor$ . Clearly  $\beta_p \leq \beta_q$ . We have  $p[1] = p[2] = \dots = p[\beta_p] > q[1] = q[2] = \dots = q[\beta_q]$ . So,  $\forall t$  such that  $1 \leq t \leq \beta_p$ ,  $\sum_{x=1}^t p[x] > \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta_p + 1 \leq t < k$ , we have  $p[t+1] = p[t+2] = \dots = p[k] = 1 \leq q[k] \leq q[k-1] \leq \dots \leq q[t+1]$ , implying  $\sum_{x=1}^t p[x] = n - \sum_{x=t+1}^k p[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ . Therefore,  $p \succeq q$ .

**Property 8** Let  $p_k^i \in \phi(n, k)$  be a major motif and  $p_k^j \in \phi(n, k)$  such that  $j > i$ . Then  $p_k^i \succeq p_k^j$ .

**Proof 6** This result is a consequence of the Property 6 and the Property 7.

Properties 6, 7 and 8 show that major motifs dominate their successors in  $\phi(n, k)$ . So, the sum coloring associated with major motif  $p_k^i$  such that  $p_k^i[1] = \alpha(G)$  is minimal in  $\phi(n, k)$  for  $G$ .

**Property 9** Let  $k$  and  $k'$  be two integers such that  $k < k'$ .

Let  $p_k^i$  be a major motif. Then  $\forall q \in \bigcup_{y=k'}^n \phi(n, y)$  such that  $q[1] \leq p_k^i[1]$ , we have  $p_k^i \succeq q$ .

**Proof 7** Let  $\beta = \left\lfloor \frac{n-k}{p_k^i[1]-1} \right\rfloor$ . Since  $p_k^i$  is major, we have  $p_k^i[1] = p_k^i[2] = \dots = p_k^i[\beta] \geq q[1] \geq q[2] \geq \dots \geq q[\beta]$ . So,  $\forall t$  such that  $1 \leq t \leq \beta$ ,  $\sum_{x=1}^t p_k^i[x] \geq \sum_{x=1}^t q[x]$ .

Moreover,  $\forall t$  such that  $\beta + 1 \leq t < k$ , we have  $p_k^i[t+1] = p_k^i[t+2] = \dots = p_k^i[k] = 1 \leq q[k] \leq q[k-1] \leq \dots \leq q[t+1]$  ( $q$  is a sequence of non-increasing positive integers),

implying  $\sum_{x=1}^t p_k^i[x] = n - \sum_{x=t+1}^k p_k^i[x] \geq n - \sum_{x=t+1}^k q[x] = \sum_{x=1}^t q[x]$ . Therefore,  $p_k^i \succeq q$ .

Property 9 shows that a major motif  $p_k^i$  dominates its successors  $q$  in  $\phi(n)$  such that  $q[1] \leq p_k^i[1]$ . Then, the sum coloring associated with major motif  $p_k^i$  such that  $p_k^i[1] = \alpha(G)$  is minimal in  $\phi(n)$  for  $G$ .  $\Sigma(p_k^i)$  is a lower bound for  $\Sigma(G)$ , named  $LBM\Sigma$ . Based on Definition 6,  $LBM\Sigma$  can be computed by Equation 8.

$$LBM\Sigma = \alpha(G) \times \frac{\beta \times (\beta + 1)}{2} + (n - \beta \times \alpha(G) - (k - \beta - 1)) \times (\beta + 1) + \sum_{j=\beta+2}^k j \quad (8)$$

In our results, we used  $\chi(G)$  as the value for  $k$  if it is known and the best known lower bound of  $\chi(G)$  otherwise. If  $\alpha(G)$  is unknown then we used an upper bound of  $\alpha(G)$ .

Graph	$ V $	$\alpha^*$	$k^*$	$LB_{Kok}$	$LBM\Sigma$	$T_{LBM\Sigma}$	$LB_{Best}$	$T_{LB_{Best}}$	$UB_{Best}$
DSJC1000.1	1000	(175)	20	1190	<b>3480</b>	1540	2762	5193	8991
DSJC1000.5	1000	15	87	1105	<b>33835</b>	408	6708	155	37594
DSJC1000.9	1000	6	215	24005	<b>85235</b>	223	26557	2741	103464
DSJC125.1	125	34	5	135	<b>297</b>	0	247	24	973
DSJC125.5	125	10	16	245	<b>851</b>	0	549	2040	1012
DSJC125.9	125	4	43	1028	<b>2108</b>	0	1691	1128	2503
DSJC250.1	250	44	4	256	<b>840</b>	117	570	2940	970
DSJC250.5	250	12	26	575	<b>2745</b>	0	1287	3936	3210
DSJC250.9	250	5	71	2735	<b>6651</b>	24	4312	N/A	8277
DSJC500.1	500	(85)	12	566	<b>1746</b>	1039	1250	1269	2841
DSJC500.5	500	13	13	578	<b>9867</b>	16	2923	3936	10897
DSJC500.9	500	5	123	8025	<b>25581</b>	84	11053	4116	29869
flat1000-50-0	1000	20	50	2225	<b>25500</b>	263	6601	118	25500
flat1000-60-0	1000	17	60	2770	<b>29914</b>	296	6640	414	30100
flat1000-76-0	1000	15	76	3850	<b>33880</b>	466	6632	98	37167
flat300-20-0	300	15	20	490	<b>3150</b>	0	1531	4506	3150
flat300-26-0	300	12	26	625	<b>3901</b>	0	1548	4212	3966
flat300-28-0	300	12	28	678	<b>3906</b>	0	1547	3750	4238
le450-15c	450	41	15	555	<b>2705</b>	1602	2610	3438	3487
le450-15d	450	41	15	555	<b>2705</b>	2266	2628	3294	3504
le450-5a	450	(94)	5	460	<b>1310</b>	2430	1193	4044	1350
le450-5b	450	(96)	5	460	<b>1290</b>	1083	1189	4020	1350
le450-5c	450	90	5	460	<b>1350</b>	2	1278	4008	1350
le450-5d	450	90	5	460	<b>1350</b>	2	1282	4296	1350
myciel3	11	5	4	17	<b>20</b>	0	17	0	21
myciel4	23	11	5	33	<b>41</b>	0	34	0	45
myciel5	47	23	6	62	<b>81</b>	0	70	0	93
myciel6	95	47	7	116	<b>158</b>	0	142	18	189
myciel7	191	95	8	219	<b>308</b>	0	286	144	381

Table 2: Our new lower bound  $LBM\Sigma$  of the chromatic sum for some instances of the *DSJC*, *flat*, *le450* and *myciel* families of graphs and the runtime  $T_{LBM\Sigma}$  to compute  $LBM\Sigma$ , compared with the results  $LB_{Kok}$  of (Kokosiński and Kwarciany 2007), the best known lower bound  $LB_{Best}$  in the literature and the runtime  $T_{LB_{Best}}$  to compute  $LB_{Best}$ . When the independence number  $\alpha^*$  of a graph cannot be obtained in reasonable time, an upper bound between parentheses was used to compute  $LBM\Sigma$ .  $UB_{Best}$  is the best known upper bound of the chromatic sum in the literature and  $k^*$  is the best known lower bound of  $\chi(G)$ .

## Empirical Results and Analysis

In this section, we present our lower bound  $LBM\Sigma$  on the instances of the literature. To compute a maximum stable set of a graph  $G$ , we work on the complement graph  $\bar{G}$  and compute a maximum clique of  $\bar{G}$  by running the state-of-the-art exact MaxClique algorithm IncMaxCLQ (Li, Fang, and Xu 2013). The test set is composed of DIMACS (Dimacs) and COLOR (Color02) graphs. The experimental results are obtained on a processor Intel Westmere Xeon E7-8837 (2.66GHz) under Linux.

Table 2 shows the results of  $LBM\Sigma$ . For each graph, we denote by *Graph* its name,  $|V|$  the number of vertices,  $\alpha^*$  the cardinality of a maximum stable set of  $G$  when this one is known, an upper bound between brackets otherwise,  $k^*$  the best known lower bound for the chromatic number  $\chi(G)$ ,  $LB_{Kok}$  the results according to (Kokosiński and Kwarciany 2007),  $LBM\Sigma$  the result of our new algebraic bound,  $T_{LBM\Sigma}$  the time expressed in seconds to compute  $LBM\Sigma$  (including the time of the maximum stable

set searching). Column  $LB_{BEST}$  gives the best known lower bound of  $\Sigma(G)$  in the literature (Jin and Hao 2016; Moukrim et al. 2010; Qinghua and Jin-Kao 2013), all are based on a partition of the graph into cliques and  $T_{LB_{Best}}$  is time in seconds to compute  $LB_{BEST}$ .  $UB_{BEST}$  is the best known upper bound of  $\Sigma(G)$  (Jin and Hao 2016; Sghiouer 2011).

The results in Table 2 focus on instances of the DIMACS and COLOR benchmarks where  $LBM\Sigma$  improves results of the literature. Among the 70 tested instances,  $LBM\Sigma$  gives an improved value of the lower bound of MSCP for 29 instances and reaches the best known lower bound  $\Sigma(G)$  in the literature for 13 instances. In particular,  $LBM\Sigma$  proves the optimal solution for *MSCP* for the instances *flat1000-50-0*, *flat300-20-0*, *le450-5c* and *le450-5d* for the first time in the literature, because the lower bound  $LBM\Sigma$  is equal to the best known upper bound for these graphs. However, for 28 instances our algebraic bound  $LBM\Sigma$  is less effective.

The following observations can be made from Table 2:

- The heuristic methods of the literature are generally based on a partition of  $G$  into cliques (Moukrim et al. 2010; Qinghua and Jin-Kao 2013) : if  $G$  can be partitioned into  $r$  cliques  $C_1, C_2, \dots, C_r$ , then  $\Sigma(G) \geq \sum_{i=1}^r \frac{|C_i| \times (|C_i| + 1)}{2}$ . These methods can give a lower bound of  $\Sigma(G)$  near the optimum for some graphs. In this case, it is hard for any other method to give a better lower bound.
- Our approach is based on searching a maximum stable set of  $G$ , which can be very hard for large sparse graphs. In this case, an upper bound of the independence number can be used to compute  $LBM\Sigma$ .
- For other graphs  $G$ , such as random graphs *DSJC* and *flat*, whose independence number  $\alpha(G)$  is known or can be efficiently computed using IncMaxCLQ, our lower bound  $LBM\Sigma$  gives good results. Indeed, although the maximum stable set problem is a NP-hard, we observe that the time needed to find a maximum stable and to compute  $LBM\Sigma$  is comparable with the time of the methods based on partition of graph into cliques.

## Conclusion

We present in this paper a new lower bound  $LBM\Sigma$  for the Minimum Sum Coloring Problem (MSCP). We first remind the notion of motifs, that is an abstraction for the solutions of MSCP. Next, we re-state a previous algebraic lower bound for MSCP in terms of motifs. Then, we extend and improve such a lower bound by focusing on the major motifs and using a structural property of a graph  $G$ , i.e., the independence number  $\alpha(G)$ . This one is computed using the efficient IncMaxCLQ solver on the complement graph  $\bar{G}$ . Finally, we evaluate this approach by comparing our results with those of the literature, showing that  $LBM\Sigma$  gives substantially better lower bound of the chromatic sum for some families of graphs in the famous benchmarks *DIMACS* and *COLOR*. In particular,  $LBM\Sigma$  allows to reach optimal MSCP solutions for four graphs in these benchmarks for the first time in the literature.

## References

Bar-Noy, A.; Bellare, M.; Halldórsson, M. M.; Shachnai, H.; and Tamir, T. 1998. On chromatic sums and distributed resource allocation. *Information and Computation* 140(2):183–202.

Benlic, U., and Hao, J.-K. 2012. A study of breakout local search for the minimum sum coloring problem. In *Simulated Evolution and Learning*. Springer. 128–137.

Bonomo, F., and Valencia-Pabon, M. 2009. Minimum sum coloring of  $p$  4-sparse graphs. *Electronic Notes in Discrete Mathematics* 35:293–298.

Bonomo, F., and Valencia-Pabon, M. 2014. On the minimum sum coloring of  $p$  4-sparse graphs. *Graphs and Combinatorics* 30(2):303–314.

Chandy, K. M., and Misra, J. 1984. The drinking philosophers problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 6(4):632–646.

Color02. <http://mat.gsia.cmu.edu/color/instances.html>.

Dimacs. <ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color>.

Jin, Y., and Hao, J.-K. 2016. Hybrid evolutionary search for the minimum sum coloring problem of graphs. *Inf. Sci.* 352-353:15–34.

Jin, Y.; Hao, J.-K.; and Hamiez, J.-P. 2014. A memetic algorithm for the minimum sum coloring problem. *Computers & Operations Research* 43:318–327.

Kokosiński, Z., and Kwarciany, K. 2007. On sum coloring of graphs with parallel genetic algorithms. In *International Conference on Adaptive and Natural Computing Algorithms*, 211–19.

Kubicka, E., and Schwenk, A. J. 1989. An introduction to chromatic sums. In *Proceedings of the 17th conference on ACM Annual Computer Science Conference*, 39–45. ACM.

Lecat, C.; Li, C.-M.; Lucet, C.; and Li, Y. 2015. Exact methods for the minimum sum coloring problem. In *Doctoral Program of Constraint Programming (CD-DP'15)*, 61–69.

Lecat, C.; Lucet, C.; and Li, C.-M. 2016. Sum coloring : New upper bounds for the chromatic strength. *arXiv:1609.02726*.

Li, Y.; Lucet, C.; Moukrim, A.; and Sghiouer, K. 2009. Greedy algorithms for the minimum sum coloring problem. In *Logistique et Transports*, LT-027.

Li, C.-M.; Fang, Z.; and Xu, K. 2013. Combining maxsat reasoning and incremental upper bound for the maximum clique problem. In *Tools with Artificial Intelligence (IC-TAI), 2013 IEEE 25th International Conference on*, 939–946. IEEE.

Malafiejski, M. 2004. *Sum coloring of graphs*. Graph Coloring. American Mathematical Society. chapter 4, 55–66.

Minot, M.; Ndiaye, S. N.; and Solnon, C. 2016. An evaluation of complete approaches for the sum colouring problem. In *Douzièmes Journées Francophones de Programmation par Contraintes (JFPC 2016)*.

Moukrim, A.; Sghiouer, K.; Lucet, C.; and Li, Y. 2010. Lower bounds for the minimal sum coloring problem. *Electronic Notes in Discrete Mathematics* 36:663–670.

Qinghua, W., and Jin-Kao, H. 2013. Improved lower bounds for sum coloring via clique decomposition. *arXiv preprint arXiv:1303.6761*.

Sghiouer, K.; Yu, L.; Lucet, C.; and Moukrim, A. 2010. A memetic algorithm for the minimum sum coloring problem. In *Conférence Internationale en Recherche Opérationnelle (CIRO'2010)*.

Sghiouer, K. 2011. *Problème de la somme coloration : approches heuristiques et bornes inférieures*. Ph.D. Dissertation, Université de Compiègne.

Wang, Y.; Hao, J.-K.; and Glover, F. 2013. Solving the minimum sum coloring problem via binary quadratic programming. *arXiv preprint arXiv:1304.5876*.