Optimisation pour l'ordonnancement et le spatial

Laurent Houssin

Habilitation à Diriger des Recherches

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *07/10/2022* **par :**

LAURENT HOUSSIN

Optimisation pour l'ordonnancement et le spatial

JURY

ROBERTO MONTEMANNI	Université de Modène	Rapporteur
CÉLINE GICQUEL	Université Paris Saclay	Rapporteur
ANDRÉ ROSSI	Université Paris-Dauphine	Rapporteur
ALAIN HAÏT	ISAE-SUPAERO	Président du Jury
RONALD MCGARVEY	IESEG	Examinateur
SOUROUR ELLOUMI	ENSTA	Examinateur
CHRISTIAN ARTIGUES	LAAS-CNRS	Examinateur

École doctorale et spécialité :
EDSYS : ECOLE DOCTORALE SYSTEMES - ED 309
Unité de Recherche :
ISAE-SUPAERO
Parrain de recherche :
CHRISTIAN ARTIGUES

Optimisation pour l'ordonnancement et le spatial

Laurent Houssin

7 octobre 2022

À Cécile
À Raphaël et Lisa

Remerciements

En premier lieu, je tiens à remercier Céline Gicquel, Sourour Elloumi, André Rossi, Roberto Montemanni, Ronald McGarvey, Christian Artigues et Alain Haït d'avoir accepté de participer au jury de cette habilitation.

Je suis particulièrement reconnaissant envers Christian, qui a accepté de parrainer cette HDR et grâce à qui j'ai pu organiser ma recherche à travers notre collaboration fructueuse. Merci à lui pour son accueil à mon arrivée à Toulouse et son amitié sans faille. J'adresse aussi des remerciements appuyés à mes directeurs de thèse, Sébastien Lahaye et Jean-Louis Boimond pour m'avoir fait confiance à cette époque.

Durant ma carrière, j'ai connu trois laboratoires : le LISA (qui s'appelle désormais LARIS), le LAAS-CNRS et l'ISAE. Surtout, j'ai eu la chance de trouver à chaque fois des conditions très favorables pour s'épanouir, tant au niveau scientifique qu'au niveau humain. J'ai découvert la recherche au LISA dans un environnement idéal, puis j'ai passé 14 formidables années au LAAS-CNRS et enfin j'ai été très bien accueilli au sein de l'ISAE. Je remercie tous les membres de ces laboratoires pour avoir su créer des conditions propices à la recherche, malgré des charges d'enseignement souvent peu compatibles avec ces activités.

Ces recherches ne sont, bien entendu, pas le fruit de mon seul travail. Je souhaite également remercier tous les collègues, doctorants et stagiaires qui ont permis la réalisation de cet accomplissement.

Je remercie aussi infiniment Cécile pour son soutien quotidien indéfectible et son enthousiasme contagieux.

Je remercie enfin ma famille qui m'a toujours soutenu et qui continue à le faire.

Table des matières

Introduction 1

I Optimisation pour l'ordonnancement 5

1 Problèmes d'ordonnancement cyclique 7
1.1 Introduction . . . 7
1.2 Problème d'ordonnancement cyclique de base . . . 8
1.2.1 Modélisation du BCSP . . . 11
1.2.2 Caractérisation du temps de cycle optimal . . . 13
1.2.3 Résolution d'un BCSP . . . 14
1.3 Problème du job shop cyclique . . . 15
1.3.1 Résoudre un problème de CJSP . . . 20
1.4 Conclusion . . . 20

2 Optimisation robuste et ordonnancement 23
2.1 Introduction . . . 23
2.2 Approche statique . . . 24
2.2.1 Ensembles d'incertitude . . . 27
2.3 Optimisation robuste discrète . . . 30
2.4 Approche multi-niveaux . . . 30
2.4.1 Incertitude sur le membre de droite . . . 31
2.5 Ordonnancement et robustesse . . . 32
2.6 Conclusion . . . 35

3 Problème d'ordonnancement cyclique de base robuste 37
3.1 Introduction . . . 37
3.2 Définition du problème $\mathcal{U}^{\Gamma}$-BCSP . . . 38
3.3 Résultats théoriques . . . 41
3.4 Approches de résolution pour le $\mathcal{U}^{\Gamma}$-BCSP . . . 43
3.4.1 Algorithme itératif . . . 43
3.4.2 Recherche binaire . . . 43
3.4.3 Adaptation de l'algorithme de Howard . . . 43
3.5 Expérimentations numériques . . . 45
3.6 Conclusion . . . 46

4 Problème du job shop cyclique robuste 47
4.1 Introduction . . . 47
4.2 Définition du problème . . . 48
4.3 Approches de résolution pour le $\mathcal{U}^{\Gamma}$-CJSP . . . 50

4.3.1 Algorithme de Branch-and-Bound pour le $\mathcal{U}^{\Gamma}$-CJSP 51
4.3.2 Méthodes de décomposition pour $\mathcal{U}^{\Gamma}$-CJSP 53
4.4 Expérimentations numériques 56
4.5 Conclusion 57

5 Modèles et algorithmes pour le problème du job shop cyclique flexible 61
5.1 Introduction 61
5.2 Définition du problème 62
5.3 Approches de résolution 63
5.3.1 Modèle mathématique 63
5.3.2 Méthode de décomposition 64
5.3.3 Méthodes approchées 66
5.4 Expérimentations numériques 67
5.5 Conclusion 67

II Optimisation pour le spatial 71

6 Allocation de fréquence pour les systèmes de satellite SDMA 73
6.1 Introduction 73
6.2 Système de télécommunication SDMA 74
6.2.1 Description des contraintes du système 76
6.2.2 Description des scénarios 76
6.3 Résolution et résultats expérimentaux. 77
6.3.1 Scénario 1 77
6.3.2 Scénario 2 77
6.4 Méthodes avec décentrage de faisceaux. 80
6.4.1 Description de la méthode 80
6.4.2 Résultats expérimentaux 82
6.5 Conclusion 84

7 Modèle d'ordonnancement pour le problème d'affectation de fréquence 85
7.1 Introduction 85
7.2 Modèles d'ordonnancement pour les problèmes d'affectation de plage de fréquence 86
7.3 Expérimentations numériques 87
7.3.1 Intérêt de la formulation en clique maximale 87
7.4 Conclusion 88

8 Optimisation de la charge utile dans les satellites de télécommunication 91
8.1 Introduction 91
8.2 Linéarisation de la norme euclidienne 92
8.2.1 Encadrement de la norme euclidienne 92
8.2.2 Qualité de l'approximation 93
8.2.3 Utilisation pour le problème du $k-$centre continu 94

8.3 Application au problème de placement de faisceaux. 94
8.4 Amélioration de la formulation 98
8.5 Conclusion 99

Conclusion et perspectives **103**

Bibliographie **113**

Table des figures

1.1 Graphe uniforme associé à l'instance de l'exemple 1. 10
1.2 Ordonnancements périodiques suivant des hauteurs différentes. 11
1.3 Graphe uniforme associé à l'instance de l'exemple 2. 12
1.4 Un ordonnancement périodique optimal associé à l'instance du BCSP de l'exemple 2. 15
1.5 Contrainte de précédence pour un job donné $\mathcal{J}_j$. 16
1.6 Graphe disjonctif associé à l'instance de l'exemple 5. 17
1.7 Un ordonnancement périodique associé à l'instance de CJSP avec $WIP = 1$. 17
1.8 Un ordonnancement périodique associé à l'instance de CJSP avec $WIP = 2$. 18

3.1 Graphe uniforme associé à l'exemple 7. 41
3.2 Temps d'exécution moyens pour R-HOW avec des budgets d'incertitude différents. 45

4.1 Graphe de précédence associé à l'exemple 8. 49
4.2 Pour $\Gamma = 0$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3. 50
4.3 Pour $\Gamma = 1$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3. 51
4.4 Pour $\Gamma = 2$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3. 51
4.5 Schéma général de l'algorithme de décomposition. 54
4.6 Nombre d'instances résolues pour les différentes approches du $\mathcal{U}^{\Gamma}$-CJSP. 59

5.1 Un ordonnancement réalisable de l'exemple 9. 63
5.2 Ordonnancement produit par le modèle (5.1). 64
5.3 Un poulet sans tête. 65
5.4 Ordonnancement produit par la décomposition de Benders. 66
5.5 Ordonnancement produit par l'heuristique basée sur la réduction de la flexibilité. 67
5.6 Gap moyen entre la solution optimale et la solution générée par la procédure heuristique. 68

6.1 Représentation 3D d'un faisceau d'antenne. 75
6.2 Représentation 2D des faisceaux d'antenne. 75
6.3 Lien utilisateurs vers passerelle. 76
6.4 Nombre moyen d'utilisateurs acceptés pour chaque nombre d'utilisateurs. 78
6.5 Gap moyen pour chaque nombre d'utilisateurs. 79

6.6 Un exemple d'affectation de fréquence. 81
6.7 Amélioration de l'affectation des fréquence après un décentrage des faisceaux. 81
6.8 Nombre moyen d'utilisateurs réaffectés et temps de calcul par réaffectation pour différentes configurations de l'algorithme. 82
6.9 Nombre moyen d'utilisateurs acceptés avant et après la procédure de décentrage en fonction du nombre d'utilisateurs pour différents algorithmes de la phase 1. 83
6.10 Nombre moyen d'utilisateurs acceptés avant et après la procédure de décentrage pour différents algorithmes de la phase 1. 84

7.1 Un graphe d'interférence. 86
7.2 Nombre d'optima moyen en fonction du pourcentage de cliques maximales utilisé avec 500 utilisateurs. 87
7.3 Nombre d'optima moyen en fonction du pourcentage de cliques maximales utilisé avec 1000 utilisateurs. 88
7.4 Makespan en fonction du nombre d'utilisateurs. 89
7.5 Nombre d'optima en fonction du nombre d'utilisateurs. 89
7.6 Makespan en fonction du nombre d'utilisateurs. 90
7.7 Nombre d'optima en fonction du nombre d'utilisateurs. 90

8.1 (a) Discrétisation des directions du plan euclidien pour $n_{\text{directions}} = 8$. (b) Approximation du disque $\mathcal{D}$ du plan euclidien par les polygones réguliers $\mathcal{P}$ et $\mathcal{P}'$ à $n_{\text{directions}}$ côtés. 93
8.2 (a-b) Évolution de l'aire des polygones approximants $\mathcal{P}$ et $\mathcal{P}'$ par rapport à l'aire d'un disque $\mathcal{D}$ de rayon unitaire. 94
8.3 (a) Une solution au problème du $k-$centre avec $N = 59$, $K = 9$, $n_{\text{directions}} = 12$. (b) Une solution au problème du $k-$centre avec $N = 144$, $K = 30$, $n_{\text{directions}} = 12$ 95
8.4 Architecture standard d'un satellite de télécommunications multifaisceaux. 96
8.5 (a),(c),(e) Allocation de réflecteurs : 117 faisceaux réguliers, 50 faisceaux réguliers, 46 faisceaux irréguliers. (b),(d),(f) Charge par faisceau : 117 faisceaux réguliers, 50 faisceaux réguliers, 46 faisceaux irréguliers (bleu= faible, rouge= forte). 97
8.6 (a) Influence du nombre de directions $n_{\text{directions}}$ sur la qualité des solutions. (b) Exemple d'une instance avec 200 utilisateurs résolue optimalement avec $n_{\text{directions}} = 12$. 98
8.7 Partitionnement en k-moyennes de la zone de service (k = 50) et domaines admissibles pour les centres. 100
8.8 Évolution du nombre de (a) variables, (b) contraintes des différents programmes linéaires suivant le nombre de clusters. 101
8.9 Comparaison du modèle linéaire mixte de placement de faisceaux basique aux modèles avec clustering exact et heuristique. 101
8.10 L'ordonnancement sans attente $J_1 \rightarrow J_2 \rightarrow J_3$. 108

8.11 L'ordonnancement sans attente $J_1 \rightarrow J_2 \rightarrow J_3$ avec une tâche de durée prolongée. Un "a" à l'intérieur d'une tâche signifie que la tâche a été avancée. 108
8.12 Taux de couverture sur un horizon de 80000s. 111

Liste des tableaux

1.1 Les durées des tâches génériques de l'instance de l'exemple 2. 12
1.2 Les données de l'instance de l'exemple 5. 17

3.1 Données de l'instance décrite dans l'exemple 7. 41
3.2 Temps d'exécution moyens en secondes pour BS, NCC and R-HOW et les valeurs de pourcentage de déviation des temps de cycle optimaux par rapport aux valeurs nominales des temps de cycle. 46

4.1 Données de l'exemple 8. 49
4.2 Trois ordonnancement de l'exemple 8 et leur pire temps de cycle pour différentes valeurs de Γ. 50
4.3 Temps d'exécution moyens en secondes, pourcentage de la valeur de déviation du temps de cycle optimal par rapport au temps de cycle nominal ainsi que le nombre d'instances résolues parmi 20 pour l'algorithme de Branch-and-Bound (B&B), l'algorithme de génération différée de contraintes (RG) et l'algorithme de génération de colonnes et de contraintes (RCG). 57
4.4 Nombre d'instances résolues en moins de 900 secondes parmi 20 pour l'algorithme de Branch-and-Bound (B&B), l'algorithme de génération différée de contraintes (RG) et l'algorithme de génération de colonnes et de contraintes (RCG) 58

5.1 Un exemple de FCJSP 62
5.2 Comparaison du temps moyen (sec) de résolution entre le MIP et les heuristiques. 69

6.1 Nombre d'optima trouvés par le MIP. 79

8.1 Valeur du rayon pour plusieurs linéarisation avec 100, 120 et 130 stations. 96
8.2 Exemple de flowshop sans attente. 107

Introduction

Parcours

Mon parcours de chercheur, ou plus précisément d'enseignant-chercheur, a débuté avec mon doctorat à Angers au sein de l'ISTIA (et qui s'appelle désormais Polytech Angers) dans un laboratoire qui se nommait le LISA (et qui est devenu le LARIS). J'ai soutenu en 2006 ma thèse qui portait sur la commande des systèmes dynamiques à événements discrets. J'ai ensuite rejoint le LAAS-CNRS et l'Université Paul Sabatier en 2007 comme maître de conférences. J'ai passé 14 (excellentes) années au sein de l'équipe MOGISA qui est ensuite devenue l'équipe ROC. Je me suis assez rapidement intéressé aux problèmes d'ordonnancement, notamment les problèmes périodiques qui présentaient ce coté "dynamique". En parallèle, le tissus industriel toulousain a également nourri mes recherches. Par l'intermédiaire de contrats industriels dans un premier temps, puis de thèses CIFRE, l'optimisation pour le spatial a alors constitué un second axe de recherche. En 2021, j'ai rejoint l'ISAE-Supaero en tant qu'enseignant chercheur dans le cadre d'un détachement. Plus précisément, j'ai intégré l'équipe SD du département DISC. L'équipe a pour objectif de fournir des outils d'aide à la décision en utilisant des techniques issues des communautés de l'intelligence artificielle et de la recherche opérationnelle.

Encadrement doctoral et scientifique

Encadrement de doctorat

- **Carla Juvin**
  Méthodes hybrides et robustes pour l'ordonnancement disjonctif avec flexibilité de ressources et contraintes complexes
  Taux d'encadrement : L. Houssin (50%), P. Lopez (50%)
  Carla a débuté sa thèse en octobre 2020. La thèse porte sur la synthèse de méthode hybride (mêlant programmation par contraintes et programmation mathématique) pour la résolution de problème d'ordonnancement disjonctif qui présente des caractéristiques qui le rendent difficile à résoudre telles que la flexibilité des ressources, la présence de fenêtre de temps pour l'exécution des tâches ou encore des temps de préparation (set-up) sur les machines entre deux tâches différentes. Carla a publié 2 communications dans des conférences internationales et un article est actuellement soumis dans une revue.

- **Idir Hamaz**
  Méthodes d'optimisation robuste pour les problèmes d'ordonnancement cyclique
  Taux d'encadrement : L. Houssin (90%), S. Cafieri (10%)
  Idir a débuté sa thèse en octobre 2015 et a soutenu le 3 décembre 2018. La

thèse portait sur le problème d'ordonnancement cyclique robuste (avec et sans contrainte de ressource). Nous avons tout d'abord démontré la complexité de la version robuste du problème d'ordonnancement de base. Puis dans un second temps, nous avons proposé trois méthodes pour la résolution du problème du jobshop cyclique robuste. Son travail a donné lieu a 2 publications en revue et 6 publications dans des conférences internationales. De plus, une autre revue est actuellement en préparation. Idir est actuellement Post-Doc au LIRMM (Montpellier).

- **Jean-Thomas Camino**
  Co-optimisation charge utile satellite et système télécom
  Taux d'encadrement : L. Houssin (50%), C. Artigues (50%)
  Jean-Thomas a débuté sa thèse en octobre 2013 et a soutenu le 22 juin 2017. La thèse portait sur le problème d'optimisation de charge utile satellite et système télécom. Il s'agissait d'une thèse CIFRE avec l'entreprise AIRBUS SATELLITE (alors appelée ASTRIUM à l'époque du début de la thèse). Son travail a donné lieu a 2 publications en revue et 3 publications dans des conférences internationales. A la suite de sa thèse Jean-Thomas a été embauché par AIRBUS SATELLITE où il occupe un poste d'ingénieur de recherche.

- **Touria Ben Rahhou**
  Nouvelles méthodes pour les problèmes d'ordonnancement cyclique
  Taux d'encadrement : L. Houssin (100%)
  Touria a effectué sa thèse sur la période octobre 2009 - juin 2013 (incluant un congés de maternité). L'objectif de la thèse était le développement de nouvelles méthodes pour les problèmes d'ordonnancement cyclique. Un Branch and Bound a été développé pour résoudre le jobshop cyclique. Durant sa thèse, Touria a publié 3 communications dans des conférences internationales. Touria est actuellement gérante de sa propre société de conseil.

- **Kata Kiamanaroj**
  Frequency allocation in SDMA satellite telecommunication systems
  Taux d'encadrement : L. Houssin (50%), C. Artigues (50%)
  Kata a débuté sa thèse en aout 2009 et il a soutenu en juin 2012. Cette thèse faisait partie d'un contrat entre ASTRIUM (Airbus Satellite) et l'agence spatiale thaïlandaise GISTDA (Geo-Informatics and Space Technology Development Agency). Le sujet portait sur les problèmes d'allocation de fréquences qui apparaissent dans les systèmes de communication par satellite. Ces problèmes correspondent en fait à des variantes de la version classique du problème d'affectation de fréquences. Deux types d'interférences ont été considérés dans cette étude : les interférences binaires et les interférences cumulatives. Nous avons montré que dans certaines conditions, ces problèmes pouvaient se ramener à des problèmes d'ordonnancement classiques. Des modèles de programmation linéaire en nombre entier et programmation par contraintes ont été implémentés pour

résoudre ces problèmes. Pendant cette période, Kata a publié 2 articles en revue et 4 communications dans des conférences internationales. Kata est maintenant ingénieur de recherche au GISTDA (équivalent du CNES en Thaïlande).

Encadrement de stage

Durant ces dernières années, j'ai encadré de nombreux stagiaires (entre 15 et 20) de niveau M1 ou M2. La liste n'est pas présentée ici. Ces stages ont généralement porté sur mes thématiques de recherche mais ils m'ont aussi permis d'initier des collaborations avec d'autres équipes.

Introduction

Ce document est une synthèse de mon activité de recherche dans le domaine de la recherche opérationnelle mais il ne présente pas l'intégralité des travaux que j'ai pu mener depuis ma thèse. J'ai choisi de mettre l'accent sur une partie que je pense représentative de mes démarches et de ma capacité à développer un programme de recherche cohérent sur la durée. Le titre de ce manuscrit représente fidèlement mes thématiques de recherche. Je m'intéresse, sans être circonscrit à ces domaines, à l'optimisation pour l'ordonnancement et à l'optimisation pour les activités spatiales.

Ce document se divise en parties :

— Optimisation pour l'ordonnancement.

— Optimisation pour le spatial.

Dans la première partie, le premier chapitre expose la problématique de l'ordonnancement cyclique. Deux problèmes sont étudiés en profondeur : le problème basique d'ordonnancement cyclique et le job shop cyclique.

Le chapitre 2 est dédié à l'optimisation robuste. Nous présentons deux contextes décisionnels qui sont le contexte statique et le contexte multi-niveaux. Puis nous présentons une vue d'ensemble de l'optimisation discrète robuste. Enfin, la dernière section de ce chapitre est consacrée à un état de l'art de la robustesse dans les problèmes d'ordonnancement.

Dans le chapitre 3, nous proposons une approche d'optimisation robuste pour le problème d'ordonnancement cyclique de base. Nous définissons l'ensemble d'incertitude considéré, ensuite nous présentons le problème que nous allons étudier. Quelques résultats théoriques concernant la caractérisation du temps de cycle optimal ainsi que les bornes sur le temps de cycle optimal au cas robuste sont exposés. Ensuite, nous proposons trois méthodes de résolution pour le BCSP robuste. Enfin, pour comparer et montrer l'efficacité de nos algorithmes, nous présentons quelques résultats numériques obtenus à partir d'instances générées d'une manière aléatoire.

Le chapitre 4 est structuré comme suit : après un rappel de la définition du CJSP et présenté l'ensemble d'incertitude, nous proposons la formulation du CJSP robuste dans le cadre de l'optimisation robuste bi-niveaux. Afin de résoudre le problème, nous proposons trois approches. La première approche est un algorithme de Branch-and-Bound. Pour les deux autres approches de résolution, nous adaptons deux méthodes

de décompositions pour les problèmes robustes bi-niveaux. Enfin, nous présentons les résultats des expérimentations numériques.

Les travaux présentés dans le chapitre 5 portent sur le problème du job shop cyclique flexible. Ce problème correspond à un problème de job shop cyclique où les machines sont flexibles. En d'autres termes, chaque tâche générique possède un sous-ensemble de machines sur lesquelles elle peut être exécutée. Cela ajoute un niveau de décision par rapport au problème classique, puisqu'en plus de déterminer l'ordre des tâches exécutées sur les machines ainsi que les dates de début, il faut déterminer l'affectation des tâches aux machines.

La seconde partie de ce manuscrit est dédiée à l'optimisation pour le spatial.

Le chapitre 6 étudie l'intérêt de la technologie SDMA pour les satellites de télécommunication. Combiné à une gestion optimisée de la ressource radio, nous montrons que cette technologie permet un net gain en nombre d'utilisateurs par rapport à une gestion plus classique de la ressource.

Dans le chapitre 7, nous exploitons les liens entre un problème d'ordonnancement et le problème d'affectation de plage de fréquence dans une constellation de satellites. Plusieurs formulations de ce problème sont testées.

Une nouvel encadrement est proposé pour la norme euclidienne dans le 8. Après avoir présenté cette linéarisation et démontré son acuité sur un problème de $k-$centres, nous l'appliquons à un problème réel de placement de faisceau dans un système de satellites. Au prix d'une approximation paramétrable, cette linéarisation autorise la présence de contraintes de proximité et de contraintes d'éloignement dans ce type de problème.

Première partie

Optimisation pour l'ordonnancement

Chapitre 1

Problèmes d'ordonnancement cyclique

Sommaire

1.1 Introduction 7
1.2 Problème d'ordonnancement cyclique de base 8
1.2.1 Modélisation du BCSP 11
1.2.2 Caractérisation du temps de cycle optimal 13
1.2.3 Résolution d'un BCSP 14
1.3 Problème du job shop cyclique 15
1.3.1 Résoudre un problème de CJSP 20
1.4 Conclusion 20

1.1 Introduction

L'ordonnancement consiste à placer des tâches dans le temps et dans l'espace. Plus précisément, cela consiste à affecter une date de début aux tâches et une ressource sur laquelle la tâche est réalisée. Ces tâches sont généralement soumises à différentes contraintes. Parmi les plus communes, les *contraintes de précédence* expriment le fait qu'une tâche ne peut pas s'exécuter avant que l'exécution d'une autre tâche soit achevée. Les *contraintes de ressource* expriment le caractère disjonctif d'une ressource. Par ressource, nous entendons des ressources humaines comme les opérateurs dans une ligne d'assemblage, ou techniques comme les machines ou les robots. Ces contraintes définissent la quantité d'une ressource qui peut être allouée à l'exécution d'une tâche donnée. On parle de ressource *disjonctive* dans le cas d'une ressource qui ne peut exécuter qu'une seule tâche à la fois. Les ressources *cumulatives* peuvent être utilisées par plusieurs tâches simultanément.

À ces contraintes d'ordonnancement se rajoute en général une (ou plusieurs) fonction(s) objectif(s) permettant de mesurer la performance des solutions possibles. Parmi les fonctions objectifs les plus utilisées, nous citons la minimisation du makespan, c'est-à-dire la durée totale de l'ordonnancement, la minimisation de la somme pondérée des dates de fin d'exécution des tâches, la minimisation du nombre de dates d'échéance non respectées, *etc*.

Usuellement, les problèmes d'ordonnancement considère que chaque tâche est exécutée une seule et unique fois. Cependant, ce chapitre s'intéresse à un autre type de

problème d'ordonnancement où l'exécution des tâches, dites *génériques*, doit être répétées une infinité de fois. Ce cas est régulièrement rencontré, par exemple, dans les cellules robotisées [Levner 2007]. En effet, un ensemble de robots exécute les mêmes tâches d'une manière répétitive, ce qui permet d'établir un ordonnancement qui va être répété continuellement. L'aspect cyclique apparaît également dans les problèmes de planning. Par exemple, les passages des transports publics sont périodiques, l'emploi du temps du personnel du transport peut l'être aussi [Liebchen 2007]. Dans d'autres cas, l'introduction d'un aspect cyclique est une des manières de réduire la taille des problèmes d'ordonnancement classique. Ceci est réalisé en créant des *batchs* qui peuvent être exécutés d'une manière cyclique.
L'ordonnancement cyclique met généralement en jeu des objectifs différents de ceux considérés en ordonnancement classique. En l'occurrence, l'objectif le plus utilisé est la minimisation du *temps de cycle* (aussi appelé *période* de l'ordonnancement). Cela revient à minimiser la différence de temps entre deux exécutions successives. Notons que minimiser le temps de cycle d'un ordonnancement est semblable à la maximisation du taux de production. Un second critère utilisé dans le cadre de l'ordonnancement périodique est la minimisation du *work-in-process*, qui représente le nombre maximum d'occurrences du même job qui peuvent être exécutées simultanément. Minimiser cette valeur revient à minimiser les stocks intermédiaires.

L'ordonnancement cyclique trouve des applications dans différents domaines. Par exemple, il existe des applications dans le domaine de la robotique [Levner 2007, Dawande 2005, Kats 2002], le pipeline logiciel [Benabid 2011, Gasperoni Asperoni 1994, Sucha 2008, Hanzalek 2011], les systèmes de production [Hanen 1997, HITZ 1979], la logistique, *etc*.

Dans ce chapitre, nous considérons les problèmes d'ordonnancement cyclique dans leurs versions déterministes. Autrement dit, tous les paramètres liés aux problèmes (comme par exemple la durée des taches) sont supposés être connus d'une manière précise. Nous décrivons deux principaux problèmes d'ordonnancement cyclique. Dans la section 1.2, nous considérons le problème d'ordonnancement cyclique de base, et dans la section 1.3 le problème du job shop cyclique, qui correspond au problème précèdent complété par des contraintes de ressource. Nous définissons formellement les deux problèmes et présentons différentes modélisations associées. Ensuite, quelques résultats théoriques concernant les conditions de faisabilité sont reportés ainsi que la caractérisation d'une solution optimale. Les différentes méthodes de résolution présentes dans la littérature ainsi que les extensions possibles de ces deux problèmes sont discutées.

1.2 Problème d'ordonnancement cyclique de base

Le problème d'ordonnancement cyclique de base (BCSP : Basic Cyclic Scheduling Problem) est un problème central en ordonnancement cyclique, étant donné qu'il constitue la base de résolution de la plupart des problèmes d'ordonnancement cyclique. En effet, dans plusieurs problèmes d'ordonnancement cyclique, une fois que le valeurs de certaines variables de décision sont fixées, le sous-problème qui en résulte est un

BCSP, ce qui permet d'utiliser les algorithmes de résolution du BCSP afin d'évaluer des solutions de problèmes plus complexes.

Soit $\mathcal{T} = \{1, ..., n\}$ un ensemble de n tâches dites génériques. Chaque tâche $i \in \mathcal{T}$ possède une durée d'exécution p_i et doit être exécutée sans préemption. Afin de distinguer les différentes exécutions de ces tâches génériques, nous notons $\langle i, k \rangle$ la k-ème occurrence de la tâche i telle que $k \in \mathbb{N}$, et $\mathsf{t}(i, k) \in \mathbb{R}$ sa date de début. Un ordonnancement cyclique σ est une affectation des dates de début $\mathsf{t}^\sigma(i, k) \in \mathbb{R}$ pour chaque occurrence $\langle i, k \rangle$ de chaque tâche générique $i \in \mathcal{T}$.

Définition 1 (**Temps de cycle asymptotique**). *Le temps de cycle asymptotique d'un ordonnancement cyclique σ est donné comme suit :*

$$\alpha^\sigma = \lim_{k \to \infty} \frac{\max \{t^\sigma(i, k) + p_i | i \in \mathcal{T}\}}{k},$$

avec p_i la durée d'exécution de la tâche générique i.

Définition 2. *Le taux de production associé à un ordonnancement σ est donné par :*

$$\tau^\sigma = \frac{1}{\alpha^\sigma},$$

où α^σ est le temps de cycle associé à l'ordonnancement cyclique σ.

Définition 3 (**Ordonnancement périodique**). *Un ordonnancement σ est dit périodique avec un temps de cycle α^σ s'il existe un vecteur $t = (t_1, ..., t_n)$ tel que*

$$t^\sigma(i, k) = t_i + \alpha^\sigma(k - 1) \quad , \forall i \in \mathcal{T}, \forall k \geq 1 \tag{1.1}$$

Notons que, pour tout i dans $\mathcal{T}$, t_i représente la date d'exécution de la première occurrence de la tâche i, soit $t^\sigma(i, 0)$.

Une des propriétés des ordonnancements périodiques est qu'un ordonnancement donné σ est totalement défini par :

— le vecteur des dates de début des premières occurrences $t = (t_i), i \in \mathcal{T}$

— le temps de cycle α^σ.

Un ordonnancement de ce type consiste à exécuter les premières occurrences des tâches génériques suivant le vecteur de dates de début t et à répéter cet ordonnancement toutes les α^σ unités de temps.

Les différentes tâches génériques sont soumises a des contraintes de précédence $\mathcal{P}$ dites aussi contraintes *uniformes*, qui peuvent être exprimées comme suit :

$$\mathsf{t}(i, k) + p_i \leqslant \mathsf{t}(j, k + H_{ij}), \quad \forall (i, j) \in \mathcal{P}, \ \forall k \geq 1, \tag{1.2}$$

Ces contraintes sont caractérisées par deux paramètres. Le premier paramètre p_i est la durée de la tâche i, souvent appelé *longueur* dans la littérature, et le second est la *hauteur* H_{ij}. Ce dernier représente le décalage d'occurrence entre les exécutions des tâches i et j. En résumé, ces contraintes expriment le fait que, pour tout $k \in \mathbb{N}$, la

$(k + H_{ij})$-ème occurrence de la tâche j ne puisse s'exécuter avant la fin de l'exécution de la k-ème occurrence de la tâche i. Étant donné que nous considérons dans ce travail des ordonnancements périodiques, en remplaçant l'expression $\mathsf{t}(i, k)$ de (1.1) dans (1.2), nous obtenons les contraintes suivantes :

$$t_j - t_i + \alpha H_{ij} \geqslant p_i, \quad \forall (i, j) \in \mathcal{P} \tag{1.3}$$

Nous ramenons donc l'ensemble infini des contraintes (1.2) à un nombre fini de contraintes définies par (1.3).

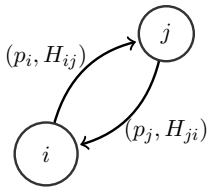


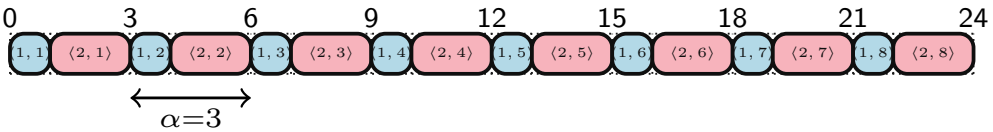
FIGURE 1.1 – Graphe uniforme associé à l'instance de l'exemple 1.

Notons que le paramètre de hauteur H_{ij} est un élément clé dans la compréhension des problèmes d'ordonnancement cyclique. En ordonnancement classique, les contraintes de précédence portent sur des tâches, tandis qu'en ordonnancement cyclique, les contraintes de précédence portent sur des occurrences de tâche générique. Le paramètre de hauteur d'une contrainte de précédence détermine quelle occurrence doit attendre la fin d'exécution d'une autre occurrence. L'exemple suivant illustre l'effet des paramètres de hauteur sur un ordonnancement.

Exemple 1. *Considérons une instance avec deux tâches génériques et deux contraintes de précédence dont les relations de précédence sont représentées dans la figure 1.1 avec $i = 1$ et $j = 2$. L'exécution de la tâche générique 1 dure 1 unité de temps et celle de la tâche générique 2 dure 2 unités de temps. Nous affectons des valeurs différentes aux hauteurs puis nous représentons les ordonnancements associés à la période $\alpha = 3$ dans la figure 1.2. Les valeurs des hauteurs sont données comme suit :*

- *Dans le diagramme de Gantt (a) $H_{12} = 0$ et $H_{21} = 1$.*
- *Dans le diagramme de Gantt (b) $H_{12} = 2$ et $H_{21} = -1$.*
- *Dans le diagramme de Gantt (c) $H_{12} = -1$ et $H_{21} = 2$.*

(a)



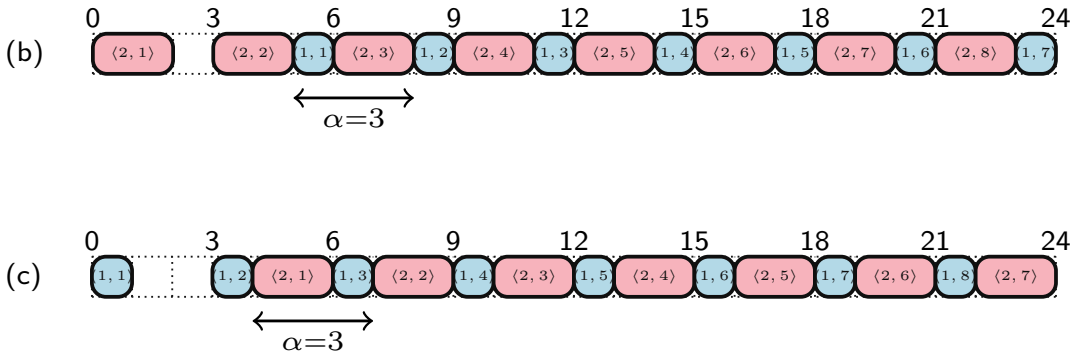


FIGURE 1.2 – Ordonnancements périodiques suivant des hauteurs différentes.

1.2.1 Modélisation du BCSP

Différents modèles ont été proposés dans la littérature pour la représentation des problèmes d'ordonnancement cyclique. Nous rappelons quelques modélisations, en nous concentrant davantage sur celles basées sur la théorie des graphes et sur la programmation linéaire.

Algèbre $(\max, +)$

La représentation par l'algèbre $(\max, +)$ a été proposée pour le BCSP dans [Cuninghame-Green 1962]. Cette approche algébrique, créée comme une alternative à l'algèbre usuelle, permet une représentation "linéaire" des systèmes mettant en jeu des phénomènes de synchronisation et de parallélisme tels que les réseaux informatiques, les systèmes de transport ou les systèmes de production. Cette approche n'est pas détaillée dans ce manuscrit mais on peut mentionner [Singh 2014], [Houssin 2011a], [Hanen 1994] et [Barman 2018].

Réseau de Petri

Un BCSP peut être modélisé par un réseau de Petri ([Benabid-Najjar 2012] et [Song 1998]). Plus précisément, par une classe de réseaux de Petri appelée graphe d'événements temporisés (TEGs : Timed event graphs). Notons que le passage d'un graphe uniforme à un TEG et l'inverse peut se faire d'une manière aisée. Les TEGs représentent un outils de modélisation et de visualisation pour des problèmes d'ordonnancement en général et des problèmes d'ordonnancement cyclique en particulier. Concernant la résolution, le graphe TEG est mis en équations avant de résoudre le problème.

Théorie des graphes

En théorie des graphes, deux représentations possibles du BCSP existent. La première est une représentation *étendue*, c'est à dire un graphe contenant toutes les oc-

currences des tâches génériques et tous les arcs de précédence. Dans ce graphe, qui est un graphe infini, chaque nœud représente une occurrence $\langle i, k\rangle$ d'une tâche générique $i \in \mathcal{T}$. Il existe un arc (i, j) entre l'occurrence $\langle i, k\rangle$ et $\langle j, k+H_{ij}\rangle$ si et seulement si il y a une contrainte de précédence générique entre i et j ayant une hauteur H_{ij}. L'autre possibilité est de représenter le BCSP par un graphe générique bi-valué $G = (\mathcal{T}, \mathcal{P}, L, H)$, appelé *graphe uniforme*. Chaque nœud $v \in \mathcal{T}$ représente une tâche générique dans le BCSP et chaque arc $e_{ij} \in \mathcal{P}$ représente une contrainte de précédence dans le BCSP. La fonction $L : A \mapsto \mathbb{N}$ associe, pour chaque arc $e_{ij} \in \mathcal{P}$, une *longueur* $L(e_{ij}) = p_i$ et la fonction $H : A \mapsto \mathbb{Z}$, appelée fonction de hauteur, associe pour chaque arc $e_{ij} \in \mathcal{P}$, une *hauteur* $H(e_{ij}) = H_{ij}$.

Dans le reste du manuscrit, nous considérons une représentation du BCSP à travers un graphe uniforme, avec $m = |\mathcal{P}|$ le nombre d'arcs.

Exemple 2. *Considérons une instance avec $n = 4$ tâches génériques et $m = 6$ contraintes de précédence. Les durées des 4 tâches sont données dans la Table 1.1, et le graphe uniforme associé à cette instance est donné dans la figure 1.3.*

Tâche	1	2	3	4
Durée	2	1	3	1

TABLE 1.1 – Les durées des tâches génériques de l'instance de l'exemple 2.

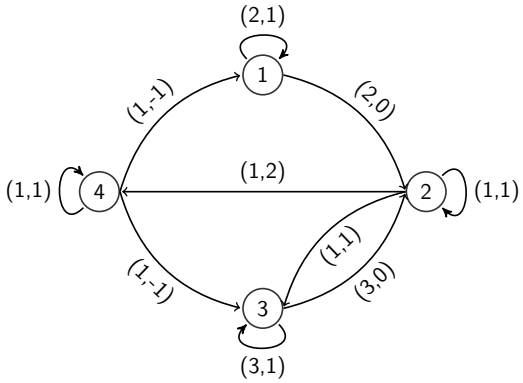


FIGURE 1.3 – Graphe uniforme associé à l'instance de l'exemple 2.

Programmation linéaire

Le problème d'ordonnancement cyclique de base peut être aussi représenté par un programme linéaire comme suit :

$$\min \quad \alpha \tag{1.4a}$$
$$s.c. \quad t_j - t_i + \alpha H_{ij} \geq p_i \quad \forall (i,j) \in \mathcal{P} \tag{1.4b}$$
$$t_i \geq 0 \quad \forall i \in \mathcal{T} \tag{1.4c}$$

L'objectif (1.4a) du problème est de minimiser le temps de cycle α. Les contraintes (1.4b) représentent les contraintes de précédence génériques. Enfin, les contraintes (1.4c) forcent les valeurs des dates de début à être positives.

1.2.2 Caractérisation du temps de cycle optimal

Dans ce qui suit, nous allons présenter les principaux résultats concernant l'existence d'un ordonnancement cyclique, ainsi que la caractérisation de la valeur du temps de cycle optimal.

Définition 4 (**Hauteur, Longueur**). *Nous définissons la hauteur d'un circuit c d'un graphe G comme $H(c) = \sum_{(i,j)\in c} H_{ij}$ et sa longueur $L(c) = \sum_{(i,j)\in c} p_i$.*

Définition 5 (**Consistance**). *Un graphe uniforme G est dit consistant si et seulement si tous les circuits ont une hauteur positive.*

Le résultat qui suit décrit une condition nécessaire et suffisante pour l'existence d'un ordonnancement périodique pour un problème d'ordonnancement cyclique de base.

Théorème 1 ([Hanen 1994]). *Soit G un graphe uniforme associé à une instance du BCSP. Il existe un ordonnancement périodique $\sigma = (t, \alpha)$ si et seulement si le graphe G est consistant.*

Définition 6 (**Temps de cycle d'un circuit**). *Le temps de cycle $V(c)$ d'un circuit c appartenant au graphe G est défini comme suit :*

$$V(c) = \frac{L(c)}{H(c)}$$

Théorème 2 ([Hanen 1994]). *Supposons qu'une instance du BCSP admette un ordonnancement périodique réalisable. Alors, la valeur du temps de cycle minimum est donnée comme suit :*

$$\alpha = \max_{c \in \mathcal{C}} V(c)$$

où $\mathcal{C}$ est l'ensemble des circuits du graphe G.

Définition 7 (**Circuit critique**). *Un circuit c est dit critique si V(c)= α.*

L'ordonnancement classique fait souvent appel à la notion de **chemin critique**. La valeur de ce chemin représente la durée minimale de l'ordonnancement et augmenter la valeur d'une tâche appartenant à ce chemin fait augmenter d'autant la durée minimale

du projet. Le circuit critique joue le même rôle que le chemin critique. Cela détermine la durée minimale du temps de cycle et augmenter la durée d'une tâche générique appartenant à ce circuit induit une augmentation de la valeur minimum du temps de cycle.

Exemple 3. *Le graphe représenté dans la figure 1.3 possède 7 circuits :* $c_1 = (1, 2, 4, 1)$, $c_2 = (2, 3, 2)$, $c_3 = (2, 4, 3, 2)$, $c_4 = (1, 1)$, $c_5 = (2, 2)$, $c_6 = (3, 3)$ *et* $c_7 = (4, 4)$. *Les ratios associés sont respectivement,* $\alpha_{c_1} = 4$, $\alpha_{c_2} = 4$ *et* $\alpha_{c_3} = 5$. *Par conséquent, le temps de cycle optimal est donné par* $\alpha = \max\{\alpha_{c_1}, \alpha_{c_2}, \alpha_{c_3}, \alpha_{c_4}, \alpha_{c_5}, \alpha_{c_6}, \alpha_{c_7}\} = 5$ *et le circuit critique associé est* $c_3 = (2, 4, 3, 2)$.

1.2.3 Résolution d'un BCSP

La résolution du BCSP peut se faire en deux temps [Hanen 1997, Chretienne 1991]. Dans un premier temps, le temps de cycle minimum est calculé. Ensuite, en utilisant un graphe particulier, les dates de début des tâches génériques sont calculées.

Différents algorithmes pour le calcul du temps de cycle minimum d'un BCSP ont été présentés dans la littérature. Parmi les algorithmes permettant le calcul du temps de cycle minimum, ou poids moyen maximum comme cela a été nommé dans divers articles, certains sont des algorithmes basés sur le graphe uniforme associé au BCSP et certains se basent sur des méthodes algébriques. Un algorithme de recherche binaire a été présenté dans [Gondran 1979] et qui a une complexité $\mathcal{O}(nm\left(log(n) + log(\max_{(i,j)\in E}(L_{ij}, H_{ij}))\right))$. Une vue globale des différents algorithmes de calcul de temps de cycle minimum peut être trouvée dans [Dasdan 2004]. Les auteurs comparent différents algorithmes et concluent que l'algorithme le plus performant, malgré sa complexité pseudo-polynomiale, est l'algorithme de Howard. Cet algorithme, présenté dans [Howard 1964], a été initialement développé pour les processus décisionnels de Markov et ensuite adapté pour les problèmes d'ordonnancement cyclique [Romanovskil 1967, Cochet-Terrasson 1998, Dasdan 1999, Dasdan 1998]. Le problème peut être résolu aussi par la théorie de l'algèbre $(\max, +)$ où le calcul du temps de cycle minimum revient à calculer la valeur propre de la matrice d'incidence du graphe associé au BCSP. Une autre manière de résoudre ce problème est d'utiliser le programme linéaire (1.4a)-(1.4c). Différentes études utilisent les réseaux de Pétri afin de modéliser le BCSP et ensuite dériver un système d'équations.

Une fois le temps de cycle optimal calculé par l'une des méthodes citées plus haut, l'ordonnancement au plus tôt peut être déterminé en calculant le plus long chemin à partir d'un nœud source s, choisi arbitrairement dans le graphe $G_\alpha = (\mathcal{T}, A)$, où à chaque arc $e_{ij} \in \mathcal{P}$ est associé un poids $a_\alpha(e_{ij}) = p_i - \alpha H_{ij}$, appelé *amplitude* [Chrétienne 1984].

Exemple 4. *Le diagramme de Gantt associé à l'ordonnancement optimal au plus tôt de l'instance du BCSP décrite dans l'exemple 2, est donné dans la figure 1.4*

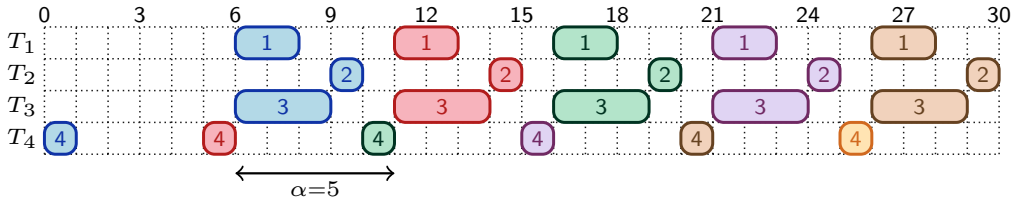


FIGURE 1.4 – Un ordonnancement périodique optimal associé à l'instance du BCSP de l'exemple 2.

1.3 Problème du job shop cyclique

Dans cette section, nous considérons le problème du job shop cyclique (CJSP : Cyclic Job Shop Problem) qui a été introduit dans [Hanen 1994]. La différence avec le BCSP est que le nombre de machines disponibles est inférieur au nombre de tâches génériques à exécuter. Par conséquent, certaines machines doivent être partagées par plusieurs tâches. Ainsi, le problème du job shop cyclique peut être considéré comme un BCSP avec des contraintes de ressources.

Dans un CJSP, chaque occurrence d'une tâche $i \in \mathcal{T} = \{1, ..., n\}$ doit être exécutée, sans préemption, sur une machine $M_{(i)} \in \mathcal{M} = \{1, ..., m\}$. Les différentes tâches génériques sont regroupées sous forme de jobs, où chaque job $j \in \mathcal{J}$ représente une séquence de tâches qui doivent être exécutées dans un ordre précis. Afin d'éviter le chevauchement entre deux exécutions de deux tâches génériques affectées sur la même machine, pour toutes les occurrences k et l $(l, k \in \mathbb{Z})$ de chaque paire de tâches i et j où $M_{(i)} = M_{(j)}$, les *contraintes disjonctives* suivantes sont introduite :

$$\mathrm{t}(i,k) + p_i \leq \mathrm{t}(j,l) \ \ \textbf{ou} \ \ \mathrm{t}(j,l) + p_j \leq \mathrm{t}(i,k) \qquad k, l \in \mathbb{Z}. \tag{1.5}$$

La contrainte (1.5) indique donc que si les tâches i et j sont exécutées sur la même machine, alors pour une occurrence k de la tache i et une occurrence l de la tâche j, il existe deux possibilités :

— soit l'occurrence k de la tache i se termine avant le début de l'occurrence l de la tâche j,

— soit l'occurrence k de la tache i débute après la fin de l'occurrence l de la tâche j.

Dans le contexte de l'ordonnancement classique, on parle parfois d'arbitrage pour ces contraintes disjonctives [Carlier 1978]. Il a été prouvé dans [Hanen 1994] que la contrainte (1.5) peut être réécrite comme suit :

$$\left.\begin{array}{l} t_j - t_i + \alpha K_{ij} \geq p_i \\ K_{ij} + K_{ji} = 1 \\ K_{ij} \in \mathbb{Z} \end{array}\right\} \quad \forall\, i, j \in \mathcal{T} : M_{(i)} = M_{(j)} \tag{1.6}$$

où K_{ij} est une variable qui représente le décalage d'occurrence entre les exécutions des

tâches i et j. Notons que, contrairement au problème du job shop classique, l'ensemble des valeurs possibles pour les variables K_{ij} est l'ensemble des entiers $\mathbb{Z}$ et non pas $\{0, 1\}$.

Pour résumer, un CJSP est défini par

- un ensemble $\mathcal{T} = \{1, ..., n\}$ de n tâches génériques,
- un ensemble $\mathcal{M} = \{1, ..., m\}$ de m machines,
- pour tout $i \in \mathcal{T}$, une tâche i ayant une durée p_i et devant être exécutée sur la machine $M_{(i)} \in \mathcal{M}$,
- un ensemble $\mathcal{D}$ de contraintes disjonctives qui sont imposées quand deux tâches sont exécutées sur la même machine,
- un ensemble $\mathcal{P}$ de contraintes de précédences,
- un ensemble $\mathcal{J}$ de jobs correspondant à une séquence de tâches génériques. Plus précisément, un job $\mathcal{J}_j$ définit une séquence $\mathcal{J}_j = O_{j,1} \dots O_{j,k}$ de k tâches génériques d'une même occurrence qui doivent être exécutées dans cet ordre.

Le CJSP peut être aussi représenté par un graphe bi-valué $G = (\mathcal{T}, \mathcal{P} \cup \mathcal{D})$ appelé *graphe disjonctif*. Les tâches génériques appartenant au même job sont liées par des arcs uniformes dans $\mathcal{P}$. Chaque arc $(i, j) \in \mathcal{P}$ est étiqueté par deux valeurs, une longueur $L_{ij} = p_i$ et une hauteur $H_{ij} = 0$. La figure 1.5 montre la représentation de contraintes de précédence d'un job donné $\mathcal{J}_j$. De plus, pour chaque paire de tâches génériques exécutées sur la même machine, une paire d'arcs (i, j) et (j, i) intervient. Ces arcs, dits *arcs disjonctifs*, sont respectivement étiquetés par $L_{ij} = p_i$ et $H_{ij} = K_{ij}$, et $L_{ji} = p_j$ et $H_{ji} = K_{ji}$ où K_{ij} et K_{ji} sont des variables de décalage d'occurrences qui doivent satisfaire $K_{ij} + K_{ji} = 1$. De plus, deux nœuds fictifs s et e représentant respectivement le début et la fin de l'ordonnancement sont rajoutés au graphe G. Pour chaque première tâche i (resp. dernière tâche j) d'un job, un arc (s, i) (resp. (j, e)) est ajouté au graphe, tel que $L_{si} = 0$ et $H_{si} = 0$ (resp. $L_{je} = p_j$ et $H_{je} = 0$). Finalement, un arc de retour allant de e à s tel que $L_{es} = 0$ et $H_{es} = WIP$, où WIP est un paramètre appelé Work-In-Process.

Ce paramètre WIP représente le nombre maximal de travaux en cours dans un même cycle. Augmenter la valeur de WIP permet d'autoriser plus d'occurrences d'un même job à un instant donné et éventuellement de diminuer la valeur du temps de cycle. Notons que si $WIP = 1$, le problème est équivalent au problème du job shop non-cyclique pour lequel l'objectif est de minimiser le makespan. A partir d'une certaine valeur du WIP, il n'est plus possible de diminuer le temps de cycle, toutefois, il n'existe pas de méthode permettant de calculer cette valeur de manière exacte. En revanche, une borne supérieure sur le WIP n'influant plus sur la valeur du temps de cycle a été proposée dans [Chauvet 2003]. L'exemple 5 permet d'illustrer ce phénomène.

FIGURE 1.5 – Contrainte de précédence pour un job donné $\mathcal{J}_j$.

Job	$\mathcal{J}_1$		$\mathcal{J}_1$	
Tâche	t_1	t_2	t_3	t_4
Durée	3	4	4	5
Machine	M_1	M_2	M_1	M_2

TABLE 1.2 – Les données de l'instance de l'exemple 5.

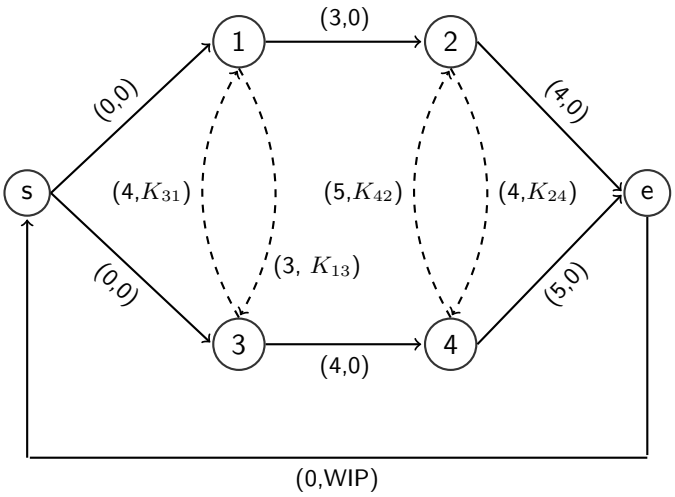


FIGURE 1.6 – Graphe disjonctif associé à l'instance de l'exemple 5.

Exemple 5. *Considérons un exemple de job shop avec un ensemble* $\mathcal{T} = \{1, 2, 3, 4\}$ *de* 4 *tâches,* 2 *jobs et* 2 *machines. Les données du problème sont décrites dans la Table 4.1. Le job* $\mathcal{J}_1$ *est composé des tâches* 1 *et* 2 *et le job* $\mathcal{J}_2$ *est composé des deux tâches restantes,* 3 *et* 4*. Le graphe disjonctif associé à ce problème est donné dans la figure 1.6. Nous voulons, à travers cet exemple, illustrer l'effet de la valeur du Work-In-Process sur l'ordonnancement. Dans un premier temps, fixons la valeur du Work-In-Process à* 1*. La solution optimale est donnée par* $K_{13} = 0$*,* $K_{31} = 1$*,* $K_{24} = 0$*,* $K_{42} = 1$ *et le temps de cycle optimal associé est* $\alpha_{wip=1} = 12$*. L'ordonnancement associé est représenté dans la figure 1.7. Maintenant, si nous augmentons la valeur du Work-In-Process à* 2*, la solution optimale devient* $K_{13} = 0$*,* $K_{31} = 1$*,* $K_{24} = 1$*,* $K_{42} = 0$ *et le temps de cycle optimal associé est* $\alpha_{opt} = 9$*. L'ordonnancement associé est représenté dans la figure 1.8.*

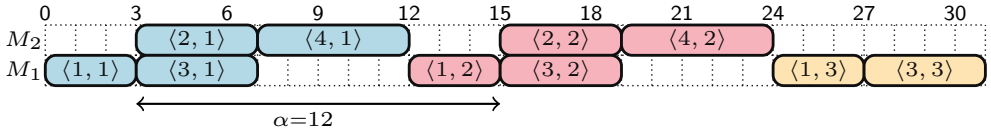


FIGURE 1.7 – Un ordonnancement périodique associé à l'instance de CJSP avec $WIP = 1$.

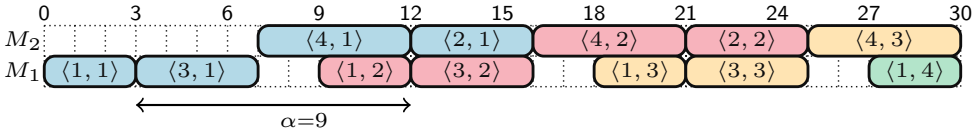


FIGURE 1.8 – Un ordonnancement périodique associé à l'instance de CJSP avec $WIP = 2$.

Résoudre un CJSP revient à trouver un vecteur $(K_{ij})_{(i,j)\in\mathcal{D}}$, c'est-à-dire un ordre d'exécution des tâches affectées sur les mêmes machines, et un vecteur des dates de début des tâches génériques de manière à ce que le temps de cycle α soit minimum.

Notons que la dominance des ordonnancements périodiques présentée dans la section 1.2 n'est plus assurée pour le CJSP. Un contre-exemple a été montré dans [Hanen 1994] pour lequel l'ordonnancement périodique n'est pas dominant.

Le CJSP peut s'exprimer sous la forme du programme mathématique suivant :

$$\min \quad \alpha \tag{1.7a}$$

$$s.c. \quad t_j - t_i + \alpha H_{ij} \geq p_i \qquad \forall (i,j) \in \mathcal{P} \tag{1.7b}$$

$$\left.\begin{array}{l} t_j - t_i + \alpha K_{ij} \geq p_i \\ K_{ij} + K_{ji} = 1 \\ K_{ij} \in \mathbb{Z} \end{array}\right\} \quad \forall\, (i,j) \in \mathcal{D} \tag{1.7c}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{T} \tag{1.7d}$$

Les variables du problème sont les dates de début des premières occurrences $(t_i)_{i\in\mathcal{T}}$, le temps de cycle α et les décalages d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$. L'objectif du problème (1.7) est de minimiser le temps de cycle α. Les contraintes (1.7b) sont des contraintes de précédence et concernent les tâches appartenant aux mêmes jobs. Les contraintes (1.7c) définissent l'ordre d'exécution des tâches affectées sur les mêmes machines. Enfin, les contraintes (1.7d) forcent les variables $(t_i)_{i\in\mathcal{T}}$ à être positives. Notons qu'une fois les variables de décalage d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$ fixées, le problème résultant est un BCSP. Par conséquent, les algorithmes de résolution du BCSP peuvent être utilisés pour évaluer une solution du CJSP.

Notons que ce problème est non-linéaire car nous avons un produit entre les variables $(K_{ij})_{(i,j)\in\mathcal{D}}$ et la variable α. Ce produit peut être linéarisé avec un changement de variable [Hanen 1997], en posant $\tau = 1/\alpha$ et $u_i = \tau \times t_i$ pour toute tâche i dans l'ensemble $\mathcal{T}$. Le problème qui résulte de ce changement de variable est un programme linéaire en variables mixtes (MIP) et peut s'écrire comme suit :

$$\max \quad \tau \tag{1.8a}$$

$$s.c. \quad u_j - u_i \geq \tau p_i - H_{ij} \qquad \forall (i,j) \in \mathcal{P} \tag{1.8b}$$

$$\left.\begin{array}{l} u_j - u_i \geq \tau p_i - K_{ij} \\ K_{ij} + K_{ji} = 1 \\ K_{ij} \in \mathbb{Z} \end{array}\right\} \quad \forall\, (i,j) \in \mathcal{D} \tag{1.8c}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{T} \tag{1.8d}$$

Les variables de décalage d'occurrence K_{ij} peuvent prendre des valeurs dans l'ensemble des entiers $\mathbb{Z}$, toutefois, uniquement un sous ensemble de valeurs dans $\mathbb{Z}$ sont réalisables. En effet, d'après le Théorème 1, les valeurs assignées aux variables de décalage d'occurrence K_{ij} ne doivent pas créer de circuits ayant une hauteur négative. Le théorème suivant caractérise une borne inférieure sur les valeurs de K_{ij} permettant de maintenir la consistance du graphe associé à une instance d'un CJSP.

Théorème 3 ([Fink 2012])**.** *Soit G un graphe associé à une instance d'un CJSP. Pour chaque variable de décalage d'occurrence K_{ij}, une borne inférieure peut être calculée comme suit :*

$$K_{ij}^{-} = 1 - \min\{H(l) \mid l \textit{ est un chemin de } j \textit{ à } i \textit{ dans } G\}. \tag{1.9}$$

Corollaire 1. *Étant donné que $K_{ij} + K_{ji} = 1$, une borne supérieure K_{ij}^{+} peut être aussi déduite :*

$$K_{ij}^{+} = 1 - K_{ij}^{-}. \tag{1.10}$$

Des bornes sur la valeur optimale du temps de cycle peuvent aussi être calculées. Une relaxation possible d'un CJSP est d'ignorer les contraintes disjonctives. Le problème engendré est un BCSP et la solution optimale associée α_{BCSP} représente une borne inférieure sur la valeur optimale du problème original. Étant donnée la structure des contraintes de précédence du CJSP, la valeur de cette borne est donnée par :

$$\alpha_{\mathrm{BCSP}} = \max_{\mathcal{J}_i \in \mathcal{J}} \frac{\sum_{j \in \mathcal{J}_i} p_j}{WIP}. \tag{1.11}$$

De plus, une autre borne inférieure peut être calculée en raisonnant sur les charges des machines. Soit $M_{(i)} \in \mathcal{M}$ une machine quelconque et $S \subseteq \mathcal{T}$ un sous-ensemble de tâches affectées sur la machine $M_{(i)}$, alors le temps de cycle optimal α est nécessairement supérieur ou égal à $\sum_{i \in S} p_i$. Comme cela est vérifié pour chaque machine $M_{(i)} \in \mathcal{M}$, nous pouvons déduire la borne supérieure suivante :

$$\alpha_{machine} = \max_{m \in \mathcal{M}} \left\{ \sum_{i \in \mathcal{T} : M_{(i)} = m} p_i \right\}. \tag{1.12}$$

Par conséquent, afin de déterminer une borne inférieure, nous pouvons prendre la valeur maximum entre ces deux bornes inférieures. La borne suivante reste toujours

valide :

$$\alpha_{lb} = \max\{\alpha_{machine}, \alpha_{\text{BCSP}}\}. \tag{1.13}$$

Exemple 6. *En appliquant les calculs de borne présentés précédemment à l'exemple 5 avec* $WIP = 2$*, nous obtenons* $\alpha_{machine} = 9$ *et* $\alpha_{BCSP} = 9/2$*. On peut remarquer que l'ordonnancement optimal avec* $WIP = 2$*, présenté en figure 1.8, atteint la borne* $\alpha_{machine}$ *.*

1.3.1 Résoudre un problème de CJSP

Peu de méthodes exactes sont présentes dans la littérature. Une des manières de résoudre le CJSP est de résoudre la formulation MIP (1.8), introduite dans [Hanen 1994], par des solveurs de programmation linéaire comme Cplex. Une méthode de Branch-and-Bound à été présentée dans [Hanen 1994]. Dans cet algorithme, les branchements sont effectués sur les décalages d'occurrence qui ne sont pas encore fixés. L'auteur utilise un branchement dichotomique. Plus précisément, à partir d'un noeud de l'arbre de recherche, le branchement est effectué sur un arc disjonctif K_{ij} et deux noeuds sont générés. Dans le premier noeud, l'intervalle des valeurs possibles pour K_{ij} est restreint à $I_{ij} = [K_{ij}^-, c_{ij}]$ et le deuxième est restreint à $[c_{ij} + 1, 1 - K_{ji}^-]$, où c_{ij} est le milieu de l'intervalle I_{ij}. Une autre méthode de Branch-and-Bound a été présentée dans [Fink 2012]. Le branchement consiste à brancher sur une variable de décalage d'occurrence $K_{ij} \in \mathcal{D}$ qui n'est pas encore fixée, et à générer un noeud fils pour chaque valeur possible dans l'intervalle I_{ij}. Pour chaque noeud généré, l'algorithme affecte une valeur différente dans I_{ij} pour K_{ij}. Une version du job shop avec des jobs identiques a été étudiée dans [Roundy 1992]. L'auteur prouve que le problème est NP-difficile et développe un algorithme de Branch-and-Bound.

Des méthodes de résolution exactes et approchées ont été proposées pour une extension du CJSP avec des contraintes de précédence linéaires. Ce problème a été étudié dans [Boussemart 2002]. Les auteurs utilisent une approche de programmation par contraintes pour résoudre le problème. Un algorithme génétique a été présenté dans [Cavory 2005] pour la résolution du CJSP avec des contraintes de précédence linéaires.

Une méthode approchée basée sur les réseaux de neurones, pour le CJSP avec minimisation du temps de cycle, a été présentée dans [Kechadi 2013].

1.4 Conclusion

Dans ce chapitre, nous avons présenté la définition d'un ordonnancement cyclique et détaillé deux problèmes d'ordonnancement cyclique. Dans la section 1.2, nous avons présenté le problème d'ordonnancement cyclique de base. Ce problème est le plus simple de cette classe d'ordonnancement étant donné qu'il possède uniquement des contraintes de précédence. Nous avons rappelé l'importance de ce problème dans la résolution d'autres problèmes d'ordonnancement cyclique plus complexes. Nous avons aussi présenté différentes manières de résoudre le BCSP et avons cité des méthodes de résolution qui

existent dans la littérature. La section 1.3 est dédiée au problème du job shop cyclique. Nous avons présenté différentes possibilités de modélisation et de résolution.

Le constat est que les problèmes d'ordonnancement cyclique sont bien étudiés dans la littérature avec différents points de vue. Plusieurs outils ont été utilisés, suivant les communautés, afin de modéliser ces problèmes et les résoudre. En revanche, la littérature de l'ordonnancement cyclique sous incertitude est très pauvre. Le Chapitre 2 présente ce paradigme, qui est l'optimisation robuste, que nous allons ensuite utiliser dans les chapitres suivants.

Chapitre 2

Optimisation robuste et ordonnancement

Sommaire

2.1 Introduction **23**
2.2 Approche statique **24**
2.2.1 Ensembles d'incertitude 27
2.3 Optimisation robuste discrète **30**
2.4 Approche multi-niveaux **30**
2.4.1 Incertitude sur le membre de droite 31
2.5 Ordonnancement et robustesse **32**
2.6 Conclusion **35**

2.1 Introduction

Dans les problèmes d'optimisation, notamment ceux issus d'applications réelles, les valeurs de certains paramètres peuvent être incertaines. Les origines de ces incertitudes sont multiples. Cela peut provenir des erreurs de mesure, d'arrondis, d'estimation ou de prévision. Ces incertitudes peuvent amener à des solutions qui sont irréalisables ou très coûteuses. En effet, une solution optimale déterministe peut devenir la pire solution en présence d'incertitudes selon le critère d'évaluation choisi. Il a été montré dans [Ben-Tal 2000a] qu'une perturbation, de l'ordre de 0.01%, des paramètres des problèmes cause plus de 50% de violation des contraintes dans 13 instances de programmes linéaires parmi 90 dans les instances de NETLIB [Ben-Tal 2002, Ben-Tal 2009]. D'où la nécessité d'une approche prenant en compte ces incertitudes et fournissant une certaine garantie quant à la faisabilité ou l'optimalité des problèmes d'optimisation.

Il existe deux approches fondamentales permettant la prise en compte de l'incertitude dans les problèmes d'optimisation : l'optimisation robuste [Ben-Tal 2000a] et la programmation stochastique [Dantzig 1955, Birge 2011]. Concernant la programmation stochastique, la distribution de probabilité des paramètres incertains est supposée être connue, et ne pas changer durant l'horizon de temps considéré. L'objectif considéré dans les problèmes de programmation stochastique est, en général, l'optimisation d'une certaine espérance. Contrairement à l'optimisation stochastique, l'optimisation robuste ne nécessite pas la connaissance de la loi de distribution des paramètres incertains, mais

uniquement d'un ensemble d'incertitudes construit à partir d'expériences ou de données historiques [Bertsimas 2009] et l'objectif considéré, en général, est l'optimisation d'un *pire cas*. L'idée est de déterminer une solution qui reste "bonne" pour tous les scénarios appartenant à l'ensemble d'incertitude décrivant les paramètres incertains. Il existe des cas où l'optimisation stochastique convient pour les problèmes d'optimisation. Chacun de ses deux paradigmes possède ses avantages et désavantages et il existe des situations où l'un convient et pas l'autre. C'est le cas par exemple quand le décideur peut être satisfait d'une garantie de probabilité. Par contre, dès qu'une certaine garantie stricte est exigée, concernant soit le coût , soit la faisabilité de la solution, alors l'optimisation robuste est un paradigme plus adapté. Le succès de l'optimisation robuste est dû à son efficacité au niveau de la résolution. En effet, nous allons voir dans la section 2.2 que les versions robustes de beaucoup de problèmes restent dans la même classe que le problème de départ ou dans une autre classe de problème soluble en pratique comme les problèmes d'optimisation quadratique.

Notons qu'il existe d'autres mesures de robustesse que le pire cas. Par exemple, le critère du *regret maximum* qui est issu de la théorie de la décision. Cela consiste à mesurer l'écart entre la valeur de la solution choisie par le décideur, et la meilleure valeur de la solution, si le décideur connaît les réalisations des paramètres incertains. L'objectif final étant de minimiser l'écart maximum. D'une manière plus formelle, cela peut s'exprimer comme suit :

$$\min_{x\in\mathcal{X}} \max_{u\in\mathcal{U}} c^u x - c^u x^*_u,$$

où $c^u x$ est la valeur de l'objectif de la solution choisie x dans le scénario u et $c^u x^*_u$ la valeur de la solution optimale pour la scénario u. Cet objectif conviendrait pour des situations où le décideur peut éprouver un certain regret en cas de mauvaise décision [Aissi 2009]. Notons que ce critère n'est pas considéré dans ce manuscrit, mais de plus amples détails peuvent être trouvés dans [Kouvelis 2013].

Dans la suite de ce chapitre, nous présentons quelques approches en optimisation robuste. Celles-ci peuvent être classées en deux catégories. La première classe concerne les problèmes d'optimisation robuste statique, dans laquelle les décisions prises ne sont pas remises en cause. La deuxième classe concerne les problèmes multi-niveaux. Plus précisément, nous allons nous focaliser sur des problèmes bi-niveaux.

2.2 Approche statique

L'optimisation robuste dans le cadre statique vise à produire une unique solution qui doit être réalisable quelque soit le scénario appartenant à l'ensemble d'incertitude décrivant les paramètres incertains. Les décisions doivent être prises avant la révélation de l'incertitude et ne peuvent pas être remises en cause. Ces décisions doivent être réalisables pour chaque scénario possible appartenant à un ensemble d'incertitudes.

Considérons le programme linéaire avec incertitude suivant :

$$\min_{x} \quad c^T x \tag{2.1a}$$
$$s.c. \quad Ax \leq b \tag{2.1b}$$

où $x \in \mathbb{R}^n$ est le vecteur de variables de décision, $c \in \mathbb{R}^n$ le vecteur des coefficients de la fonction objectif, $A = (a_j^T)_{j=1,\dots,m} \in \mathbb{R}^{m \times n}$ la matrice des coefficients des contraintes et $b \in \mathbb{R}^m$ le vecteur du membre de droite.

Nous nous intéressons au cas où certains paramètres sont incertains. Ces incertitudes peuvent porter sur les coefficients de la fonction objectif, sur la matrice des contraintes, sur le membre de droite ou une combinaison de ces paramètres. Supposons que pour le problème déterministe (2.1a)-(2.1b), tous les paramètres sont incertains, le vecteur c prend des valeurs dans C, la matrice A et le vecteur b dans l'ensemble $\mathcal{U}$. La contrepartie robuste associée au problème déterministe (2.1a)-(2.1b) peut alors s'écrire comme suit :

$$\min_{x} \max_{c \in C} \quad c^T x \tag{2.2a}$$
$$s.c. \quad Ax \leq b \quad \forall\, (A, b) \in \mathcal{U} \tag{2.2b}$$

Dans ce qui suit, nous allons nous focaliser sur des problèmes dont les incertitudes concernent uniquement la matrice des contraintes. Ceci est sans perte de généralité, étant donné que tout problème sous forme (2.2a)-(2.2b) peut être ré-écrit sous forme d'une contrepartie robuste où les incertitudes portent uniquement sur les coefficients de la matrice des contraintes [Ben-Tal 2009]. Autrement dit, toute contrepartie robuste avec des coefficients de la fonction objectif incertains et un membre de droite incertain peut être reformulée comme suit :

$$\min_{x,t} \quad t \tag{2.3a}$$
$$s.c. \quad c^T x - t \leq 0 \quad \forall\, c \in C \tag{2.3b}$$
$$Ax - b x_{n+1} \leq 0 \quad \forall\, (A, b) \in \mathcal{U} \tag{2.3c}$$

où $t \in \mathbb{R}$ est une nouvelle variable et x_{n+1} est une variable qui est égale à 1.

Dans ce qui suit, nous considérons que chaque paramètre incertain $a_{ij}(\xi)$ est une fonction affine de ξ. Plus précisément, le paramètre est défini comme suit : $a(\xi) = \bar{a} + \hat{a}\xi$ où $\bar{a}$ est la valeur nominale, $\hat{a}$ est la déviation maximale par rapport à la valeur nominale et ξ un paramètre incertain, dit *paramètre incertain primitif*, appartenant à l'ensemble Ξ. Étant donné un ensemble d'incertitude Ξ^j pour chaque contrainte j où $j = 1, \dots, m$, la *contrepartie robuste* du problème s'écrit comme suit :

$$\min_{x} \quad c^T x \tag{2.4a}$$
$$s.c. \quad (\bar{a}_j^T + \hat{a}_j^T \xi) x \leq b_j \quad \forall\, j = 1, \dots, m,\ \xi \in \Xi^j \tag{2.4b}$$

Dans cette contrepartie robuste, pour chaque contrainte j, nous avons un ensemble d'incertitudes Ξ^j. Dans le cas d'un ensemble d'incertitudes Ξ concernant toutes les contraintes, alors chaque ensemble Ξ^j, où $j = 1, \ldots, m$, correspond simplement à la projection de l'ensemble Ξ sur l'espace des données de la contrainte j [Ben-Tal 2000a].

La formulation (2.4) contient un ensemble très grand (voir infini) de contraintes qui, par conséquent, rendent le problème difficile à résoudre dans cette forme. Afin de résoudre la contrepartie robuste, il existe deux approches principales. Une première approche basée sur la re-formulation du problème (2.4) et la deuxième approche dite *approche adversiale*.

La première approche consiste à reformuler (2.4) de sorte que sa taille soit raisonnable. La première étape de cette approche consiste à éliminer le quantificateur universel "$\forall$" en utilisant le fait que les contraintes sont satisfaites pour chaque scénario $\xi \in \Xi^j$ si et seulement si les contraintes sont vérifiées dans le cas où la partie gauche prend sa valeur maximum. En d'autres termes, les contraintes (2.4b) sont satisfaites si et seulement si les contraintes suivantes sont satisfaites :

$$\bar{a}_j^T x + \max_{\xi \in \Xi^j} \left\{ \hat{a}_j^T x \xi \right\} \leq b^j \quad j = 1, \ldots, m \tag{2.5}$$

Ensuite, la contrepartie robuste finale peut être obtenue en utilisant la dualité pour le problème de maximisation dans (2.5). Suivant la nature de l'ensemble d'incertitudes considéré, nous obtenons des formulations différentes qui peuvent appartenir à des classes d'optimisation différentes. Des exemples de re-formulation vont être présentés dans la Sous-section 2.2.1.

La seconde méthode est l'approche adversiale. Elle consiste à considérer, au départ, un problème dit *problème restreint* contenant uniquement, pour chaque contrainte j, un sous-ensemble $S^j \subseteq \Xi^j$ de scénarios. Ensuite, utiliser un problème, dit *problème de séparation*, qui permet de détecter s'il existe un scénario qui n'est pas dans S^j et pour lequel la solution actuelle n'est pas faisable. À chaque itération, le problème restreint est résolu, ce qui nous fournit une solution. Cette solution est utilisée par le problème de séparation, deux cas peuvent alors se présenter :

— soit la solution associée est réalisable pour tous les scénarios et dans ce cas la solution est optimale.
— Soit, dans le cas contraire, il faut trouver un scénario pour lequel la solution actuelle est irréalisable, le rajouter à l'ensemble S^j et réitérer jusqu'à ce qu'il n'y ait plus de scénario dans Ξ causant une infaisabilité.

Notons qu'une étude expérimentale comparative entre les deux méthodes à été présentée dans [Bertsimas 2016]. Les auteurs testent les deux méthodes une fois avec un ensemble d'incertitude polyhedral [Bertsimas 2004] et une fois avec un ensemble d'incertitude ellipsoïdale [Ben-Tal 1999, El Ghaoui 1997, El Ghaoui 1998]. Enfin, les auteurs utilisent des méthodes de statistique pour les comparer.

2.2.1 Ensembles d'incertitude

Il existe différentes manières de modéliser les incertitudes pouvant affecter les paramètres d'un problème d'optimisation. Comme vu précédemment, suivant l'ensemble d'incertitude, nous pouvons avoir des contreparties robustes appartenant à des différentes classes de problèmes d'optimisation. Plus précisément, certaines contreparties robustes vont être des programmes linéaires, d'autres des programmes coniques quadratiques. L'autre point intéressant concernant les ensembles d'incertitudes est la garantie de probabilité. En effet, des probabilités concernant la faisabilité des contraintes ont été présentées pour certains ensembles d'incertitude. La première étude allant dans ce sens à été présentée dans [Ben-Tal 2000b] pour l'ensemble d'incertitude ellipsoïdal.

Dans cette partie, seront abordés quelques ensembles d'incertitudes qui existent dans la littérature et montrés, pour certains de ces ensembles d'incertitudes, les formulations des contreparties robustes associées.

Ensemble d'incertitude de type budget

Cet ensemble, introduit par Bertsimas et Sim [Bertsimas 2004], est un cas particulier de l'ensemble d'incertitude polyhedral. Dans cet ensemble, chaque paramètre incertain a_{ij} de la matrice des contraintes $A = (a_j^T)_{j=1,\dots,m} \in \mathbb{R}^{m\times n}$ est modélisé par un intervalle $[\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}]$, où $\bar{a}_{ij}$ est la valeur nominale et $\hat{a}_{ij}$ la déviation maximale autorisée par rapport à la valeur nominale. De plus, pour chaque contrainte j, le nombre de paramètres autorisés à dévier de leurs valeurs nominales est limité par Γ_j. Ce paramètre, appelé *budget d'incertitude*, est choisi par le décideur. Ceci est justifié par le fait que le cas où tous les paramètres dévient de leurs valeurs nominales arrive rarement. Donc, permettre d'ajuster ce paramètre offre une certaine flexibilité aux décideurs pour choisir des solutions plus ou moins prudentes. D'une manière plus formelle, l'ensemble d'incertitude de type budget que nous allons noter Ξ^j peut être décrit comme suit :

$$\Xi^j = \left\{ \xi \in \mathbb{R}^n | \sum_i \xi_i \leq \Gamma_j, -1 \leq \xi_i \leq 1 \right\} \tag{2.6}$$

Notons que le modèle de Soyster introduit dans [Soyster 1973] est un cas particulier du modèle d'incertitude de type budget avec $\Gamma_j = n$, où n est le nombre de paramètres incertains dans le problème considéré. Nous reconsidérons la contrepartie robuste (2.4a)-(2.4b) avec un ensemble d'incertitude de type budget, ce qui donne la formulation suivante :

$$\min_x \quad c^T x \tag{2.7a}$$
$$sc. \quad \sum_i \bar{a}_{ij} x_i + \max_{\xi\in\Xi^j} \{\sum_i \hat{a}_{ij} x_i \xi_i\} \leq b_j \qquad j = 1,\dots,m \tag{2.7b}$$
$$x_i \geq 0 \qquad i = 1,\dots,n \tag{2.7c}$$

où Γ_j est le budget d'incertitude associé à la contrainte j. Les contraintes (2.7b) sont

non-linéaires à cause du problème de maximisation. Afin de palier à cela, le problème dual de ce dernier est formulé. Ainsi, une fois le problème de maximisation remplacé par son dual, nous obtenons le problème suivant :

$$\begin{aligned}
\min_{x} \quad & c^T x && && (2.8a)\\
sc. \quad & \textstyle\sum_i \bar{a}_{ij} x_i + z_j \Gamma_j + \sum_i \lambda_{ij} \leq b_j && j = 1, \ldots, m && (2.8b)\\
& z_j + \lambda_{ij} \geq \hat{a}_{ij} && i = 1, \ldots, n, j = 1, \ldots, m && (2.8c)\\
& x_i \geq 0 && i = 1, \ldots, n && (2.8d)\\
& z_j \geq 0 && j = 1, \ldots, m && (2.8e)\\
& \lambda_{ij} \geq 0 && i = 1, \ldots, n, j = 1, \ldots, m && (2.8f)
\end{aligned}$$

Notons que ce problème correspond à un programme linéaire, il peut donc être résolu facilement par les logiciels d'optimisation. Au delà du fait que la contrepartie robuste reste linéaire, des garanties de probabilité peuvent être obtenues.

Ensemble d'incertitude polyhedral

Dans ce qui suit, après avoir défini l'ensemble d'incertitude polyhedral, nous montrons que la contrepartie robuste d'un programme linéaire avec des paramètres incertains décrits par cet ensemble peut être re-formulée sous forme d'un programme linéaire. L'ensemble d'incertitude polyhedral [Ben-Tal 1999] est définit comme suit :

$$\Xi = \{\xi | D\xi \leq d\},$$

où $D \in \mathbb{R}^{m \times n}$ et $d \in \mathbb{R}^n$.

Notons que l'ensemble d'incertitude de type budget décrit précédemment est un cas particulier de l'ensemble d'incertitude polyhedral.

La contrepartie robuste est donnée comme suit :

$$\begin{aligned}
\min_{x} \quad & c^T x && && (2.9a)\\
sc. \quad & \textstyle\sum_i \bar{a}_{ij} x_i + \max_{\xi : D\xi \leq d} \{\sum_i \hat{a}_{ij} x_i \xi_i\} \leq b_j && j = 1, \ldots, m && (2.9b)\\
& x_i \geq 0 && i = 1, \ldots, n && (2.9c)
\end{aligned}$$

En calculons le dual du problème de maximisation dans la contrainte (2.9c) nous obtenons le problème suivant :

$$\begin{aligned}
\min_{x} \quad & d_i^T \pi_i && && (2.10a)\\
sc. \quad & D_i^T \pi_i \geq \textstyle\sum_i \hat{a}_{ij} x_i && j = 1, \ldots, m && (2.10b)\\
& x_i \geq 0 && i = 1, \ldots, n && (2.10c)
\end{aligned}$$

En substituant le problème de maximisation dans la contrainte (2.9c) par son dual

(2.10a)-(2.10c), nous obtenons le problème suivant :

$$\min_{x} \quad c^T x \tag{2.11a}$$
$$sc. \quad \sum_i \bar{a}_{ij} x_i + d_i^T \pi_i \leq b_j \qquad j = 1, \ldots, m \tag{2.11b}$$
$$D_i^T \pi_i \geq \sum_i \hat{a}_{ij} x_i \qquad j = 1, \ldots, m \tag{2.11c}$$
$$x_i \geq 0 \qquad i = 1, \ldots, n \tag{2.11d}$$

La contrepartie robuste associée à un programme linéaire avec des paramètres incertains modélisés par un ensemble d'incertitude polyhedral est encore un programme linéaire.

Ensemble d'incertitude ellipsoïdal

L'ensemble d'incertitude ellipsoïdal est considéré dans [Ben-Tal 1999, El Ghaoui 1997, El Ghaoui 1998]. Chaque paramètre incertain est exprimé avec une fonction affine $a_{ij} = \bar{a}_{ij} + \hat{a}_{ij}\xi_i$, où $\bar{a}_{ij}$ est la valeur nominale et $\hat{a}_{ij}$ la valeur de la déviation par rapport à la valeur nominale et $\xi_i \in [-1, +1]$. Pour chaque contrainte j, un paramètre Ω_j permet de contrôler le degré de robustesse. En effet, ce paramètre peut être fixé par le décideur et permet d'exclure les valeurs extrêmes des intervalles qui peuvent être jugées peu probables. Cet ensemble d'incertitude est donné comme suit :

$$\Xi^j(\Omega_j) = \left\{ \xi \in \mathbb{R}^n | \sqrt{\sum_i \xi_i^2} \leq \Omega_j \right\}$$

La contrepartie robuste associée à cet ensemble d'incertitude s'écrit comme suit :

$$\min_{x} \quad c^T x \tag{2.12a}$$
$$sc. \quad \sum_i \bar{a}_{ij} x_i + \max_{\xi : \sqrt{\sum_i \xi_i^2} \leq \Omega_j} \{\sum_i \hat{a}_{ij} x_i \xi_i\} \leq b_j \qquad j = 1, \ldots, m \tag{2.12b}$$
$$x_i \geq 0 \qquad i = 1, \ldots, n \tag{2.12c}$$

Le dual du problème de maximisation dans la contrainte (2.12b) est donné comme suit :

$$\min_{x} \quad \Omega_i z_i \tag{2.13a}$$
$$sc. \quad \sum_i \bar{a}_{ij}^2 x_j^2 \leq z_i^2 \qquad j = 1, \ldots, m \tag{2.13b}$$
$$x_i \geq 0 \qquad i = 1, \ldots, n \tag{2.13c}$$
$$x_i \geq 0 \qquad i = 1, \ldots, n \tag{2.13d}$$

En substituant le problème de maximisation dans la contrainte (2.9c) par son dual

(2.13a)-(2.13c), nous obtenons le problème suivant :

$$\min_{x} \quad c^T x \tag{2.14a}$$
$$sc. \quad \sum_i \bar{a}_{ij} x_i + \Omega_i z_i \leq b_j \qquad j = 1, \ldots, m \tag{2.14b}$$
$$\sum_i \bar{a}_{ij}^2 x_j^2 \leq z_i^2 \qquad j = 1, \ldots, m \tag{2.14c}$$
$$x_i \geq 0 \qquad i = 1, \ldots, n \tag{2.14d}$$

La contrepartie robuste (2.14a)-(2.14d) est un problème non-linéaire mais quadratique et correspond à un programme conique du second ordre. Avec cette approche, pour des paramètres incertains ξ ayant une distribution symétrique dans $[-1, 1]$, la probabilité qu'une contrainte i soit violée n'excède pas $exp(-\Omega_i/2)$ [Ben-Tal 2000b].

2.3 Optimisation robuste discrète

Les problèmes d'optimisation discrète sont des problèmes où les décisions sont des variables discrètes et non pas continues. Ces problèmes sont en général NP-difficiles, bien qu'il existe certains cas particuliers polynomiaux. Dans [Bertsimas 2003], les auteurs ont étudié un problème d'optimisation discrète où les coefficients de la fonction objectif sont incertains et peuvent varier dans des intervalles, et le nombre total de déviations est borné par un budget d'incertitude. Les auteurs ont montré que le problème peut être traité en résolvant au plus $n + 1$ instances de problèmes nominaux, où n est le nombre de paramètres incertains.

Ces résultats ont été étendus dans [Poss 2014]. L'auteur étudie le cas où le budget d'incertitude est défini par une fonction des variables de décision du problème. Le problème obtenu fournit des solutions moins prudentes que la version du problème avec le budget d'incertitude classique. Cette extension peut être aussi résolue en résolvant $n+1$ problèmes polynomiaux. L'auteur a aussi proposé une formulation en programmation linéaire mixte du problème. Enfin, ce dernier montre que s'il existe un programme dynamique pour résoudre le problème combinatoire dans le cas déterministe en $\mathcal{O}(N)$, alors il existe une extension de cet algorithme pour le cas avec budget d'incertitude classique, qui peut se résoudre en $\mathcal{O}(N\Gamma)$, et une extension pour le cas avec budget d'incertitude défini comme fonction des variables, qui peut se résoudre soit en $\mathcal{O}(Nn\Gamma)$ soit en $\mathcal{O}(Nn^2\Gamma)$.

Plus de détails sur des problèmes d'optimisation robuste combinatoire peuvent être trouvés dans [Buchheim 2017].

2.4 Approche multi-niveaux

L'approche d'optimisation robuste multi-niveaux a été décrite pour la première fois dans [Ben-Tal 2004]. Contrairement au cadre statique, le cadre multi-niveaux permet la remise en cause de certaines décisions. Plus précisément, d'une part, certaines variables dites "here-and-now" doivent être décidées avant la révélation de l'incertitude et ne

peuvent pas être remises en cause. D'autre part, le reste des variables dites "wait-and-see" peuvent être calculées après la révélation de l'incertitude et ainsi s'adapter au scénario réalisé. Contrairement au cadre statique, permettre à certaines variables d'être ajustées offre une certaine flexibilité et permet d'avoir des solutions moins prudentes. Ceci dit, il existe certains cas où les deux cadres sont équivalents, par exemple, quand l'incertitude affectant chaque contrainte est indépendante des incertitudes d'autres contraintes. Dans ce manuscrit, nous nous focalisons sur les problèmes d'optimisation robuste bi-niveaux. Une vue globale sur les problèmes d'optimisation robuste multi-niveaux peut être trouvée dans [Yanikoglu 2017]. La contrepartie robuste d'un programme linéaire dans le cadre muli-niveaux s'écrit comme suit :

$$\min_{x,y(.)} \quad c^T x \tag{2.15a}$$

$$s.c. \quad A(\xi)x + By(\xi) \leq b(\xi) \quad \forall \xi \in \Xi^j \tag{2.15b}$$

où $c^T \in \mathbb{R}^n$ représente le vecteur des coûts, $x \in \mathbb{R}^n$ le vecteur des variables de décisions du premier niveau, ξ le vecteur des paramètres incertains, $A(\xi) \in \mathbb{R}^{m \times n}$ représente la matrice des coefficients incertains, $y(\xi) \in \mathbb{R}^l$ représente les variables du second niveau. Ces variables sont représentées sous forme d'une fonction qui dépend du paramètre incertain ξ. Le vecteur $B \in \mathbb{R}^{m \times l}$ représente la matrice des coefficients certains et $b \in \mathbb{R}^m$ le membre de droite.

Le problème (2.15a)-(2.15b) correspond à un problème robuste multi-niveaux avec un recours fixe, c'est-à-dire, les valeurs de la matrice B sont certaines et ne dépendent pas du paramètre incertain ξ. Les coefficients de la fonction objectif sont certains mais ceci est sans perte de généralité puisque, comme nous l'avons montré pour le cas statique, tout problème avec incertitude sur les coefficients de la fonction objectif peut être réécrit sous forme d'un problème avec une incertitude uniquement sur les contraintes.

Dans la sous-section 2.2.1, nous avons montré qu'il est possible d'expliciter, pour certains ensembles d'incertitude, une re-formulation de la contrepartie robuste d'un problème d'optimisation dans le cadre statique en un problème déterministe. Ce problème peut être un programme linéaire ou un programme conique du second ordre. Notons que pour les problèmes d'optimisation robuste multi-niveaux, ce n'est plus le cas. Il est possible d'obtenir une re-formulation de la contrepartie robuste en restreignant les variables $y(.)$ à des fonctions affines. Toutefois, cette re-formulation n'est pas exacte et représente une approximation du problème initial. L'utilisation d'arbre de décision, comme dans [Quezada 2020], peut également être utilisé pour représenter l'incertitude.

2.4.1 Incertitude sur le membre de droite

Une multitude de problèmes en optimisation sont caractérisés par des incertitudes sur les paramètres du membre de droite des contraintes. Par exemple, il est possible d'avoir de l'incertitude sur la capacité dans un problème de lot-sizing. [Santos 2018] ou de sac-a-dos [Bouman 2011], ou sur la demande dans un problème de network design [Atamturk 2007], *etc.* Concernant le problème d'optimisation avec de l'incertitude sur le membre de droite, nous avons montré que ce problème correspond au problème

déterministe où tous les paramètres incertains prennent, soit leur valeurs maximum pour un problème de minimisation, soit leurs valeurs minimum pour un problème de maximisation. Cela n'est plus valable pour les problèmes d'optimisation robuste bi-niveaux. Permettre à certaines variables d'être ajustées une fois l'incertitude révélée produit des solutions moins prudentes. Ce problème robuste bi-niveaux avec incertitude sur le membre de droite peut être formulé comme suit :

$$\min_{x,y(.)} \quad c^T x \tag{2.16a}$$

$$s.c. \quad Ax + By(\xi) \leq b(\xi) \quad \forall \xi \in \Xi \tag{2.16b}$$

Ce problème est similaire à (2.15a)-(2.15b) sauf que la matrice des coefficients des contraintes $A \in \mathbb{R}^{m \times n}$, est supposée certaine.

Un problème d'optimisation robuste bi-niveaux avec un ensemble d'incertitude de type budget à été étudié dans [Thiele 2009]. Les auteurs proposent une approche de résolution qui utilise de plans sécants (*cutting planes*), basée sur la méthode de Kelley [Kelley 1960]. Ils étudient également le cas d'un recours simple et montrent que le problème peut être résolu en résolvant m problèmes linéaires déterministes où m est le nombre de contraintes. Deux approches de décomposition ont été présentées dans [Ayoub 2016]. Les deux approches fonctionnent de la même manière. Au départ, les auteurs considèrent le problème restreint, c'est-à-dire, le problème où uniquement un sous-ensemble de scénarios est considéré. Résoudre ce problème fournit une solution du premier niveau, c'est-à-dire une valeur pour $x \in \mathbb{R}^n$. Ensuite, un problème de séparation est résolu afin de déterminer si, pour chaque scénario $\xi \in \Xi$, il existe une solution de second niveau $y(\xi) \in \mathbb{R}^l$ qui est réalisable. Les deux méthodes diffèrent dans la manière de remonter cette information au problème restreint. Dans la méthode de génération de contraintes, une coupe est ajoutée au problème restreint ; dans la méthode de génération de contraintes et variables, un bloc de variables et de contraintes correspondant au scénario $\xi \in \Xi$ pour lequel la solution du premier niveau $x \in \mathbb{R}^n$ n'est pas faisable est ajouté au problème restreint. Par ailleurs, la notion de dualité en optimisation robuste a été étudiée dans [Minoux 2009]. L'auteur prouve que le dual d'un programme linéaire robuste n'est pas équivalent à la version robuste du dual du problème de départ. De plus, l'auteur étudie un problème d'ordonnancement de base où les durées des tâches sont incertaines. Il formule le problème en un problème d'optimisation robuste bi-niveaux avec incertitude sur le membre de droite. En utilisant des propriétés de la matrice, l'auteur propose une re-formulation du problème et montre enfin que celle-ci peut être résolue avec un programme dynamique en un temps polynomial.

2.5 Ordonnancement et robustesse

Les problèmes d'ordonnancement cyclique ont été étudiés de différents points de vue. Toutefois, la plupart de ces études considèrent des paramètres certains. Or, en pratique, il est rare d'avoir une bonne estimation des paramètres caractérisant le pro-

blème. Il est aussi difficile de prévoir une panne ou un quelconque aléa. Cela peut avoir des conséquences coûteuses, soit l'ordonnancement calculé n'est pas faisable soit sa performance devient très loin de l'optimum. D'où la nécessité de la prise en compte des incertitudes dans les modèles d'optimisation pour les problèmes d'ordonnancement.

À notre connaissance, une seule étude est dédiée à l'application de l'optimisation robuste dans le cadre de l'ordonnancement cyclique. Cette étude est présentée dans [Che 2015]. L'auteur étudie un problème appelé CHSP (Cyclic Hoist Scheduling Problem). Dans ce problème, des produits doivent passer par un ensemble de réservoirs. Le temps passé par un produit dans un réservoir doit être compris dans un intervalle défini. Le transport des produits d'un réservoir à l'autre se fait par le biais d'un robot et ces robots doivent partager un même rail de transport. Aussi, les déplacements de ces robots ne doivent pas produire de collision. L'incertitude considérée porte sur les durées de transport d'un réservoir à l'autre. Ce qui peut donc engendrer le non respect des contraintes,et par conséquent, créer des produits défectueux. Afin de prendre en compte cette incertitude, l'auteur définit une manière de mesurer la robustesse d'un ordonnancement et propose un programme linéaire en nombres entiers bi-objectif. Les deux objectifs concernent l'optimisation du temps de cycle et de la robustesse. Les auteurs montrent que le temps de cycle croit en fonction de la robustesse et que la formulation du problème possède une infinité de solutions pareto-optimales.

Le reste des études sur l'ordonnancement cyclique sous incertitude est basé sur des versions stochastiques. C'est-à-dire que les coefficients incertains sont décrits par une loi de probabilité. Une version du job shop cyclique avec des paramètres stochastiques a été étudiée dans [Zhang 1997]. L'auteur considère le problème avec une seule puis plusieurs machines. Les incertitudes considérées concernent les pannes des machines, ce qui peut changer le cours d'exécution de l'ordonnancement déjà prévu. Ces pannes sont modélisées par des variables aléatoires indépendantes et identiquement distribuées. L'auteur propose un formulation mathématique où l'objectif est de minimiser une somme pondérée des valeurs espérées des retards. Ce problème a été aussi étudié du point de vue des chaînes de Markov dans [Bowman 1993] . Un problème d'ordonnancement cyclique dans les lignes d'assemblage et de production avec des durées des tâches stochastiques à été étudié dans [Karabati 1998]. Les auteurs considèrent comme fonction objectif la minimisation du temps de cycle et proposent deux heuristiques afin de résoudre le problème.

Il y a un réel manque, dans la littérature, d'approches d'optimisation robuste pour l'ordonnancement cyclique. Toutefois, il existe des études concernant la prise en compte d'incertitudes dans les problèmes d'ordonnancement plus classiques en utilisant des approches d'optimisation robuste. Dans [Bougeret 2018], les auteurs étudient des problèmes d'ordonnancement où les durées des tâches sont incertaines et appartiennent à un ensemble d'incertitudes de type budget. Ils étudient un problème d'ordonnancement sur une seule machine avec une somme (pondérée ou non pondérée) des dates de fin d'exécution des tâches. Des algorithmes d'approximation pour le problème d'ordonnancement sur des machines parallèles avec minimisation du makespan ont été aussi proposés. Enfin, les auteurs prouvent que le problème d'ordonnancement sur une seule machine avec minimisation de la somme pondérée des dates de fin est un problème NP

-difficile. Le problème d'ordonnancement sur une seule machine avec minimisation de la somme pondérée des dates de fin à été étudié dans [Ales 2018]. Les auteurs considèrent deux ensembles d'incertitudes, de type budget et ellipsoïdal. Afin de résoudre ces deux problèmes respectivement, ils proposent une re-formulation en programme linéaire en nombres entiers et une re-formulation en un programme conique de second ordre et les comparent avec des algorithmes de plans sécants (Banch-and-Cut). Un problème d'ordonnancement de projet où les durées des tâches sont incertaines a été étudié dans [Artigues 2013]. Les auteurs considèrent comme objectif, la minimisation du regret maximum par rapport au makespan et proposent deux approches de résolution. Une approche exacte où un problème restreint contenant uniquement un sous ensemble de scénarios est résolu, à la suite de quoi, un sous-problème est utilisé afin de trouver un scénario pour lequel la solution actuelle n'est pas faisable. L'algorithme itère entre les deux étapes jusqu'à ce qu'il n'y ait plus de scénario causant une infaisabilité de la solution courante. La deuxième approche proposée est une approche heuristique permettant de trouver des solutions dans un délai relativement court. Le même problème avec cette fois-ci un ensemble d'incertitude de type budget et un critère de pire cas à été étudié dans [Bruni 2018, Bruni 2017]. Les auteurs proposent une méthode de décomposition afin de résoudre le problème. Un problème d'ordonnancement de projet sous contraintes de ressources multi-modes avec des durées incertaines à été étudié dans [Balouka 2018]. Afin de résoudre le problème, les auteurs proposent une approche de génération différée de contraintes. Un problème d'ordonnancement de base où les durées des tâches sont incertaines à été étudié dans [Minoux 2009]. L'auteur propose une modélisation sous forme d'un problème d'optimisation robuste bi-niveaux avec de l'incertitude sur le membre de droite et en utilisant des propriétés de la matrice, l'auteur propose une re-formulation du problème. Enfin, il montre que le problème peut être résolu avec un programme dynamique en un temps polynomial. Plus récemment, nous pouvons aussi citer les travaux de [LEV 2022] qui traitent du problème de flow shop robuste pour lequel les durée des tâches sont incertaines.

Nous avons cité quelques études concernant les méthodes d'optimisation robuste pour les problèmes d'ordonnancement mais il existe d'autre méthodologies permettant de prendre en compte les incertitudes dans les problèmes d'ordonnancement. De nombreux auteurs parlent de "robustesse" dans les problèmes d'ordonnancement. Notons que ce terme désigne la performance d'un algorithme face aux incertitudes et non pas la robustesse au sens optimisation robuste. Dans ce qui suit, nous résumons quelques méthodes permettant de prendre en compte les incertitudes dans les problèmes d'ordonnancement.

Ordonnancement réactif. L'ordonnancement réactif vise à réparer ou ré-ordonnancer un ordonnancement déjà conçu afin de prendre en compte les aléas pouvant affecter le bon déroulement de l'ordonnancement. Soit l'ordonnancement réactif agit sur l'ordonnancement avec des règles très simples, comme par exemple, la règle *right shift* qui consiste à décaler toutes les tâches affectées par la survenue de l'aléa vers la droite [Sadeh 1993]. Soit c'est tout l'ordonnancement qui est remis en cause, on parle alors de ré-ordonnancement. Dans les deux cas, il est nécessaire d'avoir un algorithme très rapide.

Ordonnancement pro-actif. L'ordonnancement pro-actif peut être assimilé à l'optimisation robuste statique. Le but de cet ordonnancement est de produire des décisions capables d'absorber le maximum d'aléas pouvant survenir durant l'exécution des tâches. Notons que ces décisions sont supposées fixes et ne peuvent pas être ajustées. Une étude sur un problème d'ordonnancement de base avec des durées incertaines a été proposée dans [Bendotti 2017]. Les auteurs considèrent la version pro-active du problème où l'objectif est de maximiser, quelque soient les valeurs des durées des tâches dans un ensemble de scénarios réels, le nombre de dates de début qui ne changent pas et montre que ce problème est polynomial. Les auteurs étudient aussi des versions réactive du problème.

Ordonnancement pro-actif-réactif [Herroelen 2005]. L'ordonnancement pro-actif-réactif combine les deux méthodologies précédentes, c'est-à-dire, l'ordonnancement pro-actif et l'ordonnancement réactif. Au départ, un ordonnancement pro-actif qui peut absorber le maximum d'aléas est construit. Ensuite, durant l'exécution des tâches, si l'ordonnancement n'arrive pas à absorber un aléa c'est à l'ordonnancement réactif de réparer le reste de l'ordonnancement, soit par une règle définie soit par le ré-ordonnancement du reste des tâches qui ne sont pas encore exécutées.

Ordonnancement basé sur le rayon de stabilité. Le rayon de stabilité est défini comme la plus grande variation possible des paramètres incertains (au sens de la norme L_2 ou L_∞) pour laquelle la faisabilité de la solution n'est pas affectée. Un problème de ligne d'assemblage où les durées des tâches peuvent dévier de leurs valeurs nominales à été étudié dans [Rossi 2016].

Ordonnancement stochastique. Dans le cadre d'ordonnancement stochastique, des paramètres incertains sont décrits par une loi de probabilité. L'objectif est de minimiser la valeur espérée du critère choisi selon la politique choisie. En effet, une politique répare l'ordonnancement selon des règles prédéfinies à partir de la survenue de l'aléa.

2.6 Conclusion

Dans ce chapitre, nous avons présenté le paradigme d'optimisation robuste. Dans la section 2.2, nous avons présenté le contexte statique. Dans ce contexte, l'objectif est de produire une et unique solution qui doit faire face aux aléas décrits par l'ensemble d'incertitude dans le pire des cas sans pouvoir remettre en cause les décisions déjà prises. Dans cette section, nous montrons aussi deux manières de résoudre la contrepartie robuste. La première manière c'est de re-formuler la contrepartie robuste. Nous avons illustré cette méthodes avec plusieurs ensembles d'incertitude. La deuxième méthode est la méthode adversiale. Dans la section 2.3, nous citons quelques travaux concernant des problèmes robustes discrets. La section 2.4 est dédiée au contexte multi-niveaux. Dans ce contexte, il existe deux types de décisions : les décisions du premier niveau qui doivent être prises avant de connaître les réalisations des paramètres incertains, et les décisions du second niveau qui doivent être prises une fois l'incertitude révélée. Nous avons souligné que les problèmes d'optimisation robuste dans ce contexte sont des problèmes extrêmement difficiles à résoudre en pratique et nous avons présenté une

méthode qui n'est pas exacte mais qui permet de contourner cette difficulté. Enfin, nous avons cité quelques travaux portant sur l'optimisation robuste et l'ordonnancement, et nous avons fini par présenter d'autres méthodologies que l'optimisation robuste permettant de prendre en compte les incertitudes. Nous avons souligné un réel manque dans la littérature concernant la prise en compte de l'incertitude dans les problèmes d'ordonnancement cyclique, et particulièrement le manque d'études concernant l'application de l'optimisation robuste aux problèmes d'ordonnancement cyclique.

Chapitre 3

Problème d'ordonnancement cyclique de base robuste

Sommaire

3.1 Introduction 37
3.2 Définition du problème $\mathcal{U}^{\Gamma}$-BCSP 38
3.3 Résultats théoriques 41
3.4 Approches de résolution pour le $\mathcal{U}^{\Gamma}$-BCSP 43
3.4.1 Algorithme itératif 43
3.4.2 Recherche binaire 43
3.4.3 Adaptation de l'algorithme de Howard 43
3.5 Expérimentations numériques 45
3.6 Conclusion 46

3.1 Introduction

Les problèmes d'ordonnancement sous incertitude ont reçu un important intérêt ces dernières années de la part de la communauté des chercheurs. En effet, un ordonnancement optimal pour un problème dans un cadre déterministe peut devenir sous-optimal voir non réalisable en présence d'incertitude. Il existe deux approches principales pour traiter ces incertitudes. D'une part, l'optimisation stochastique qui nécessite une distribution de probabilité des paramètres incertains et dont l'objectif est d'optimiser une certaine espérance. D'autre part, l'optimisation robuste, présentée dans le chapitre 2, qui nécessite uniquement un ensemble décrivant les incertitudes qui portent sur les paramètres incertains et dont l'objectif est d'optimiser un certains pire cas.

Dans ce chapitre, nous nous intéressons au problème d'ordonnancement cyclique de base, en considérant la version du problème où les durées des tâches génériques sont incertaines et appartiennent à un ensemble d'incertitude. Nous allons en particulier considérer l'ensemble d'incertitude introduit par [Bertsimas 2004]. Dans cet ensemble, les paramètres incertains, c'est-à-dire, dans notre cas, les durées des tâches, sont modélisées par des intervalles et le niveau de robustesse est contrôlé par un paramètre appelé *budget d'incertitude*. Ce paramètre est fixé par le décideur et, suivant sa valeur, nous pouvons obtenir des solutions plus ou moins prudentes. Autrement dit, au lieu de considérer que toutes les tâches peuvent dévier de leur valeur nominale, uniquement

un sous-ensemble de tâches est autorisé à dévier de leur valeur nominale. Suivant la taille du sous-ensemble, des ordonnancements différents peuvent être obtenus. Pour ce type de problème, les décisions sont réparties en deux groupes. Le groupe de décisions du premier niveau qui doivent être prises avant la réalisation de l'incertitude, et les décisions du second niveau qui sont retardées et qui doivent être décidées une fois que l'incertitude est révélée. Plus précisément, nous cherchons à trouver la plus petite valeur du temps de cycle dans le pire cas, de sorte qu'un ordonnancement cyclique existe quel que soit le scénario. Afin de résoudre ce problème, nous allons d'abord formuler un problème de séparation. Celui-ci constituera une base pour deux algorithmes de résolution Le troisième algorithme proposé est une adaptation de l'algorithme de Howard.

Le plan de ce chapitre est donné comme suit. Dans la section 3.3, nous définissons l'ensemble d'incertitude considéré, ensuite nous définissons le problème que nous allons étudier. Dans la section 3.5, nous présentons, dans un premier temps, deux algorithmes de séparation permettant de prouver la réalisabilité d'un temps de cycle donné. Ensuite, nous présentons brièvement trois méthodes de résolution possibles pour le BCSP robuste. Deux méthodes sont basées sur l'algorithme de séparation qui revient à détecter la présence de circuits de coût négatif dans un graphe particulier, et la dernière méthode est une extension de l'algorithme de Howard. Enfin, pour comparer et montrer l'efficacité de nos algorithmes, nous présentons dans la section 3.5 quelques résultats numériques obtenus à partir d'instances générées d'une manière aléatoire.

3.2 Définition du problème $\mathcal{U}^{\Gamma}$-BCSP

Considérons un ensemble $\mathcal{T} = \{1, ..., n\}$ de n tâches génériques qui doivent êtres exécutées d'une manière infinie. Chaque tâche générique $i \in \mathcal{T}$ possède une durée nominale $\bar{p}_i$. Par contre, à cause des différentes incertitudes qui peuvent survenir, par exemple, la dégradation du rendement de la machine sur laquelle la tâche est exécutée, une panne de machine ou seulement une erreur d'estimation, cette durée peut être allongée. On note ce retard $\hat{p}_i$. Donc dans l'idéal, l'exécution de la tâche i dure $\bar{p}_i$ et dans le cas d'un aléa, $\bar{p}_i + \hat{p}_i$. En plus de la limitation de retards maximums, nous limitons le nombre de tâches pouvant dévier de leurs valeurs nominales, en introduisant le paramètre Γ. Ce paramètre, appelé budget d'incertitude (voir chapitre 2), permet de contrôler le niveau de conservatisme de la solution. Cet ensemble d'incertitude a été introduit dans [Bertsimas 2004] et a connu immédiatement un grand succès grâce au fait que la contrepartie robuste peut être résolue en général d'une manière efficace.

Soit ξ_i une variable binaire égale à 1 si la tâche i dévie de sa valeur nominale et 0 sinon. Les durées des tâches incertaines $(p_i(\xi))_{i\in\mathcal{T}}$ peuvent être modélisées comme suit :

$$p_i(\xi) = \quad \bar{p}_i + \xi_i \hat{p}_i, \ \forall \xi \in \mathcal{U}^{\Gamma}, i \in \mathcal{T}$$

où

$$\mathcal{U}^{\Gamma} = \quad \left\{ (\xi_i)_{i\in\mathcal{T}} \Big| \sum_{i=1}^{\mathcal{T}} \xi_i \leq \Gamma, \ \xi_i \in \{0, 1\} \right\}$$

représente l'ensemble d'incertitude. Le paramètre Γ désigne le nombre maximum de tâches qui sont autorisées à dévier de leurs valeurs nominales. Chaque élément de $\mathcal{U}^{\Gamma}$ est donc un scénario concordant avec le budget d'incertitude Γ.

Remarque 1. *L'ensemble $\mathcal{U}^{\Gamma}$ est générique puisque chaque motif de l'ordonnancement est sujet au même ensemble d'incertitude.*

Modèle statique

Afin de montrer l'intérêt de considérer le BCSP robuste bi-niveaux, nous allons, dans un premier temps, formuler le problème sous forme d'un problème d'optimisation robuste statique. La version statique du BCSP robuste peut se formuler comme suit :

$$\min \quad \alpha \tag{3.1}$$

$$s.t. \quad t_j - t_i + \alpha H_{ij} \geq p_i(\xi) \quad \forall (i,j) \in \mathcal{P}, \forall \xi \in \mathcal{U}^{\Gamma} \tag{3.2}$$

L'objectif du problème est de trouver la plus petite valeur du temps de cycle α. Celui-ci doit garantir la faisabilité des contraintes de précédence quel que soit le scénario réalisé dans l'ensemble d'incertitude $\mathcal{U}^{\Gamma}$.

Ce problème correspond à un programme linéaire robuste avec incertitude sur la partie droite des contraintes. Si le budget d'incertitude vaut n, alors ce problème est équivalent à celui où tous les paramètres incertains prennent leur valeurs maximales, à savoir, $p_i = \bar{p}_i + \hat{p}_i$. Le problème (4.1)-(3.2) est donc équivalent au problème déterministe suivant :

$$\min \quad \alpha \tag{3.3}$$

$$s.t. \quad t_j - t_i + \alpha H_{ij} \geq \bar{p}_i + \hat{p}_i \quad \forall (i,j) \in \mathcal{P} \tag{3.4}$$

Ce problème similaire à (1.4) peut donc être résolu polynomialement par les algorithmes du BCSP déterministe décrits dans le chapitre 1. L'inconvénient des solutions produites par (3.3)-(3.4) est que les ordonnancements sont très prudents. C'est-à-dire que les solutions produites sont très pessimistes, par conséquent cela engendre une dégradation importante au niveau du temps de cycle. Afin d'avoir des solutions moins prudentes, une approche d'optimisation robuste bi-niveaux est plus appropriée.

Modèle bi-niveaux

Dans le contexte de l'optimisation bi-niveaux, nous considérons deux types de variables : des variables de premier niveau et de second niveau. L'unique variable du premier niveau correspond au temps de cycle et les variables de second niveau sont les dates de début des premières occurrences des tâches, qui sont calculées après que l'incertitude soit révélée et peuvent être ajustées afin de prendre en compte les aléas :

$$\min \quad \alpha \tag{3.5}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq p_i(\xi) \quad \forall (i,j) \in \mathcal{P}, \forall \xi \in \mathcal{U}^{\Gamma} \tag{3.6}$$

Le problème (3.5)-(3.6) contient deux groupes de variables de décision. D'un coté, nous avons une seule variable α de premier niveau, qui doit être déterminée avant que les valeurs réelles des durées des tâches soient révélées. Une fois les valeurs de ces durées connues, les dates de début des tâches génériques peuvent être ajustées et prendre en compte les incertitudes. L'objectif de ce problème est de minimiser la pire valeur du temps de cycle α parmi tous les scénarios possibles dans l'ensemble $\mathcal{U}^{\Gamma}$. Les contraintes (3.6) expriment le fait que, pour chaque scénario $\xi \in \mathcal{U}^{\Gamma}$, il doit exister un vecteur $(t_1(\xi), \ldots, t_n(\xi)) \in \mathbf{R}^n$ qui respecte les contraintes de précédence génériques.

Exemple 7. *La figure 3.1 illustre le graphe uniforme G associé à l'instance du problème d'ordonnancement cyclique de base robuste décrit dans le tableau 3.1. Contrairement au BCSP déterministe, les longueurs des arcs (i,j) appartiennent à un intervalle $[\bar{p}_i, \bar{p}_i + \hat{p}_i]$.*

Nous considérons tout d'abord le pire cas puis le cas où le budget d'incertitude vaut $\Gamma = 1$. Calculer un ordonnancement statique, c'est-à-dire calculer un vecteur des dates de début d'exécution des premières occurrences qui ne peut pas être changé au cours de l'ordonnancement, revient à considérer le cas où toutes les tâches prennent leur valeur la plus grande. Autrement dit, dans le cadre statique, le temps de cycle optimal dans le pire cas est donné par le scénario $\xi = (1,1,1,1)$ et sa valeur est 9. Maintenant, si nous considérons que les dates de débuts peuvent être ajustées et $\Gamma = 1$, il y a 5 scénarios possibles :

- *Le scénario $\xi_1 = (0,0,0,0)$: le temps de cycle est $\alpha_1 = 5$ et il est donné par le circuit critique $c_1 = (2,4,3,2)$. Dans ce scénario, aucune tâche ne dévie de sa valeur nominale.*
- *Le scénario $\xi_2 = (1,0,0,0)$: le temps de cycle est $\alpha_2 = 7$ et il est donné par le circuit critique $c_1 = (1,2,3,1)$.*
- *Le scénario $\xi_3 = (0,1,0,0)$: le temps de cycle est $\alpha_3 = 6$ et il est donné par le circuit critique $c_3 = (2,4,3,2)$.*
- *Le scénario $\xi_4 = (0,0,1,0)$: le temps de cycle est $\alpha_4 = 6$ et il est donné par le circuit critique $c_4 = (2,4,3,2)$.*
- *Le scénario $\xi_5 = (0,0,0,1)$: le temps de cycle est $\alpha_5 = 6$ et il est donné par le circuit critique $c_1 = (2,4,3,2)$.*

Donc la valeur minimale du temps de cycle dans le cas où $\Gamma = 1$ est $\alpha_{opt} = \max\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\} = 7$ et le circuit critique associé est $c_1 = (1,2,3,1)$. Le scénario réalisant cette valeur est $\xi_2 = (1,0,0,0)$.

La version non-cyclique de ce problème à été traitée dans [Minoux 2009]. L'auteur propose une formulation sous forme d'un programme linéaire robuste bi-niveaux et, en utilisant la structure de la matrice des contraintes de ce problème, l'auteur propose une reformulation en termes de chemins et montre que le problème peut se résoudre

Tâche	1	2	3	4
durée	[2,5]	[1,2]	[3,4]	[1,2]

TABLE 3.1 – Données de l'instance décrite dans l'exemple 7.

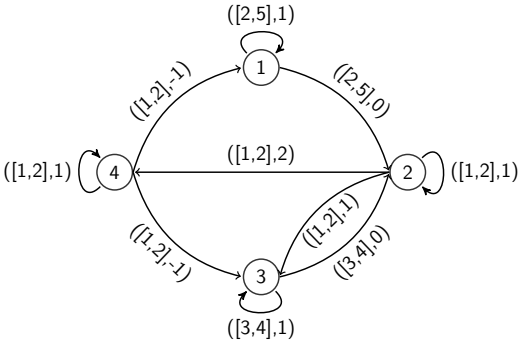


FIGURE 3.1 – Graphe uniforme associé à l'exemple 7.

en temps polynomial par un schéma de programmation dynamique. Le $\mathcal{U}^{\Gamma}$-BCSP ayant une structure différente, nous allons exploiter une des propriétés du problème afin de proposer des algorithmes de résolution appropriées pour notre problème.

3.3 Résultats théoriques

Dans cette section, nous allons étendre quelques résultats théoriques concernant la condition d'optimalité, la caractérisation du temps de cycle optimal, ainsi que l'extension des bornes sur la valeur du temps de cycle optimal. Ces résultats vont nous permettre de développer des approches de résolution qui vont être présentées dans la section 3.4.

Problème de séparation

Dans ce qui suit, nous montrons que vérifier si un temps de cycle est réalisable peut être effectué en temps polynomial. La proposition suivante caractérise une condition nécessaire et suffisante pour la réalisabilité d'un tempe de cycle donnée $\bar{\alpha}$ d'un $\mathcal{U}^{\Gamma}$-BCSP.

Proposition 1. *Un temps de cycle $\bar{\alpha}$ est réalisable si et seulement si la solution du*

programme linéaire en variables mixtes suivant

$$\max \quad \sum_{e\in E}(\bar{p}_e - \bar{\alpha}H_e)u_e + \sum_{e\in E}\hat{p}_e v_e \tag{3.7}$$

$$s.t. \quad \xi \in \mathcal{U}^{\Gamma} \tag{3.8}$$

$$\sum_{e\in\sigma^-(i)} u_e - \sum_{e\in\sigma^+(i)} u_e = 0 \quad \forall\, i \in \mathcal{T} \tag{3.9}$$

$$v_e \leq \xi_e \quad \forall\, i \in \mathcal{T} \tag{3.10}$$

$$v_e \leq u_e \quad \forall\, i \in \mathcal{T} \tag{3.11}$$

$$0 \leq u_e \leq 1 \quad \forall\, e \in \mathcal{P} \tag{3.12}$$

$$0 \leq v_e \leq 1 \quad \forall\, e \in \mathcal{P} \tag{3.13}$$

$$\xi_e \in \{0,1\} \quad \forall\, e \in \mathcal{P} \tag{3.14}$$

est non-positive.

Preuve. Reconsidérons le problème (3.5)-(3.6). Vérifier si un temps de cycle $\bar{\alpha}$ est réalisable revient à vérifier que le problème suivant possède une solution :

$$\min \quad 0 \tag{3.15}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) \geq p_i(\xi) - \bar{\alpha}H_{ij} \quad \forall (i,j) \in \mathcal{P}, \forall\, \xi \in \mathcal{U}^{\Gamma} \tag{3.16}$$

D'après le lemme de Farkas, le problème (3.15)-(3.16) admet une solution si et seulement si la valeur de l'objectif du problème suivant :

$$\max \quad \sum_{e\in E}(p_e(\xi) - \bar{\alpha}H_e)u_e \tag{3.17}$$

$$s.t. \quad \xi \in \mathcal{U}^{\Gamma} \tag{3.18}$$

$$\sum_{e\in\sigma^-(i)} u_e - \sum_{e\in\sigma^+(i)} u_e = 0 \quad \forall\, i \in \mathcal{T} \tag{3.19}$$

$$u_e \geq 0 \quad \forall\, e \in \mathcal{P} \tag{3.20}$$

est non-positive.
Notons que $\sigma^-(i)$ et $\sigma^+(i)$ représentent respectivement l'ensemble des prédécesseurs directs et l'ensemble des successeurs directs de la tâche $i \in \mathcal{T}$, et $(u_e)_{e\in\mathcal{P}}$ sont des variables duales de $(t_i)_{i\in\mathcal{T}}$.

Ce problème est non-linéaire puisqu'il y a un produit entre les variables duales $(u_e)_{e\in\mathcal{P}}$ avec les variables ξ dans la fonction objectif. Ces produit peut être linéarisé en utilisant les techniques classiques de reformulation en programmation linéaire en variables mixtes. Ceci peut être fait en introduisant une variable v_e, où $e = (i,j)$, pour chaque produit $\xi_e u_e$, et en rajoutant les contraintes (3.10) et (3.11) qui vont forcer v_e à être égale à zéro quand ξ_e ou u_e sont égaux à zéro. Ces contraintes n'ont pas de "grand M" car nous avons contraint les variables $(u_e)_{e\in\mathcal{P}}$ à être inférieures à 1. Notons que cela n'affecte en rien notre problème car nous nous intéressons uniquement au signe de

la fonction objectif et non pas à la valeur elle-même. □

3.4 Approches de résolution pour le $\mathcal{U}^{\Gamma}$-BCSP

Dans cette sous-section, nous présentons brièvement trois méthodes de résolution pour le $\mathcal{U}^{\Gamma}$-BCSP. Les deux premières méthodes sont basées sur la procédure de séparation et la dernière est une adaptation de l'algorithme de Howard [Howard 1964]. Ces méthodes de résolution sont exhaustivement décrites dans [Hamaz 2018b] et [Hamaz 2018a].

3.4.1 Algorithme itératif

Afin de résoudre $\mathcal{U}^{\Gamma}$-BCSP, nous proposons un algorithme itératif qui se base sur l'algorithme de Floyd-Warshall. Le pseudo-code associé est décrit dans l'algorithme 1.

Algorithm 1 Algorithme itératif (IA)

Step 1 : Calculer une borne inférieure α_{lb}.
Step 2 : Exécuter l'algorithme de Floyd-Warshall modifié sur $G'_{\alpha_{\text{neg}}}$.
Step 3 : **Si** Un circuit de coût négatif c est détecté **alors**
Soit ξ le scénario réalisant le circuit de coût négatif c.
Mettre à jour α_{lb} à la valeur du ratio du circuit c ($\alpha_{lb} = \frac{P_c(\xi)}{H(c)}$).
Retourner à **step 2**.
else Le temps de cycle α_{lb} est optimal.

L'algorithme d'annulation de circuits de coût négatifs a une complexité $\mathcal{O}(n^3\Gamma^2|\mathcal{C}|)$, où $|\mathcal{C}|$ est le nombre de circuit dans le graphe $G'_{\alpha_{neg}}$.

3.4.2 Recherche binaire

Une autre idée sur la résolution du $\mathcal{U}^{\Gamma}$-BCSP est d'utiliser une recherche binaire. Cette méthode consiste à diviser le domaine de recherche et réduire le domaine jusqu'à ce que la solution optimale soit trouvée. Le pseudo-code associé à notre méthode de résolution est donné dans Algorithme 2.

L'algorithme de recherche dichotomique a une complexité $\mathcal{O}(n^3\Gamma^2 \log(n\bar{p}_{max} + \Gamma\hat{p}_{max}))$, et vu que $\Gamma \leq n$, l'algorithme a une complexité polynomiale $\mathcal{O}(n^5 \log(n\bar{p}_{max} + n\hat{p}_{max}))$.

3.4.3 Adaptation de l'algorithme de Howard

L'algorithme de Howard, proposé initialement dans [Howard 1964], a été originalement développé pour définir des politiques dans les processus décisionnels markoviens. Ensuite, l'algorithme a été adapté puis amélioré pour le calcul d'un poids moyen maximum d'un graphe dans [Cochet-Terrasson 1998, Dasdan 2004]. Le principe général de l'algorithme de Howard est d'utiliser un graphe spécifique appelé *graphe de politique*. Les noeuds de ce graphe sont les mêmes que ceux du graphe uniforme associé au

Algorithm 2 Algorithme de recherche binaire (BS)

Input: Un graphe $G = (\mathcal{T}, E)$, un budget d'incertitude Γ et des poids $\bar{c} : E \mapsto \mathbb{R}$, $\hat{c} : E \mapsto \mathbb{R}$.
Output: Temps de cycle optimal

while $\alpha_{ub} - \alpha_{ub} > \varepsilon$ **do**
  $\alpha_{\mathrm{m}} = \frac{\alpha_{ub} - \alpha_{ub}}{2}$
  if pas de circuit de coût négatif **then**
    $\alpha_{ub} = \alpha_{\mathrm{m}}$
  else
    $\alpha_{lb} = \alpha_{\mathrm{m}}$
if $\alpha_{lb} > \alpha_{ub}$ **then**
  Le problème est infaisable.
return α_{m}

BCSP, sauf que chaque noeud $i \in \mathcal{T}$ ne possède qu'un seul successeur. Étant donné cette propriété, le graphe de politique possède un nombre polynomial de circuits, et par conséquent, le circuit de ratio maximum associé peut être calculé en temps polynomial par un simple algorithme de marquage. L'adaptation de l'algorithme de Howard est donnée dans l'Algorithme 3. La complexité de l'algorithme de Howard adapté est $\mathcal{O}(n^2 m \Gamma^2 |\mathcal{C}|)$.

Algorithm 3 Algorithme de Howard

Input: Un graphe $G = (\mathcal{T}, \mathcal{P})$, un budget d'incertitude Γ et des poids $\bar{c} : \mathcal{P} \mapsto \mathbb{R}$, $\hat{c} : \mathcal{P} \mapsto \mathbb{R}$.
Output: Le temps de cycle optimal.

for $i \leftarrow 1$ **to** n **do**
  for $\gamma \leftarrow 0$ **to** n **do**
    $d_i^\gamma = p_i$
  $\pi_i = i$
modifié $\leftarrow$ Vrai
while modifié $=$ Vrai **do**
  Calculer la valeur du ratio α_{lb} associé au graphe G.
  Soit h un noeud quelconque appartenant au circuit critique
  if $\exists$ un chemin de i à h dans G **then**
    $d_i^\gamma \leftarrow \max\{d_{\sigma^+(i)}^{\gamma-1} + \bar{p}_i + \hat{p}_i - \alpha_{lb} H_{i\sigma^+(i)};\ d_{\sigma^+(i)}^{\gamma} + \bar{p}_i - \alpha_{lb} H_{i\sigma^+(i)}\}$
  modifié $\leftarrow$ Faux
  for $(i,j) \in \mathcal{P}$ **do**
    for $\gamma \leftarrow 0$ **to** n **do**
      if $d_i^\gamma < \max\{d_j^{\gamma-1} + \bar{p}_i + \hat{p}_i - \alpha_{lb} H_{ij} - \varepsilon;\ d_j^\gamma + \bar{p}_i - \alpha_{lb} H_{ij} - \varepsilon\ \}$ **then**
        $d_i^\gamma \leftarrow \max\{d_j^{\gamma-1} + \bar{p}_i + \hat{p}_i - \alpha_{lb} H_{ij} - \varepsilon;\ d_j^\gamma + \bar{p}_i - \alpha_{lb} H_{ij} - \varepsilon\}$
        $\sigma^+(i) \leftarrow j$
        modifié $\leftarrow$ Vrai
return α_{lb}

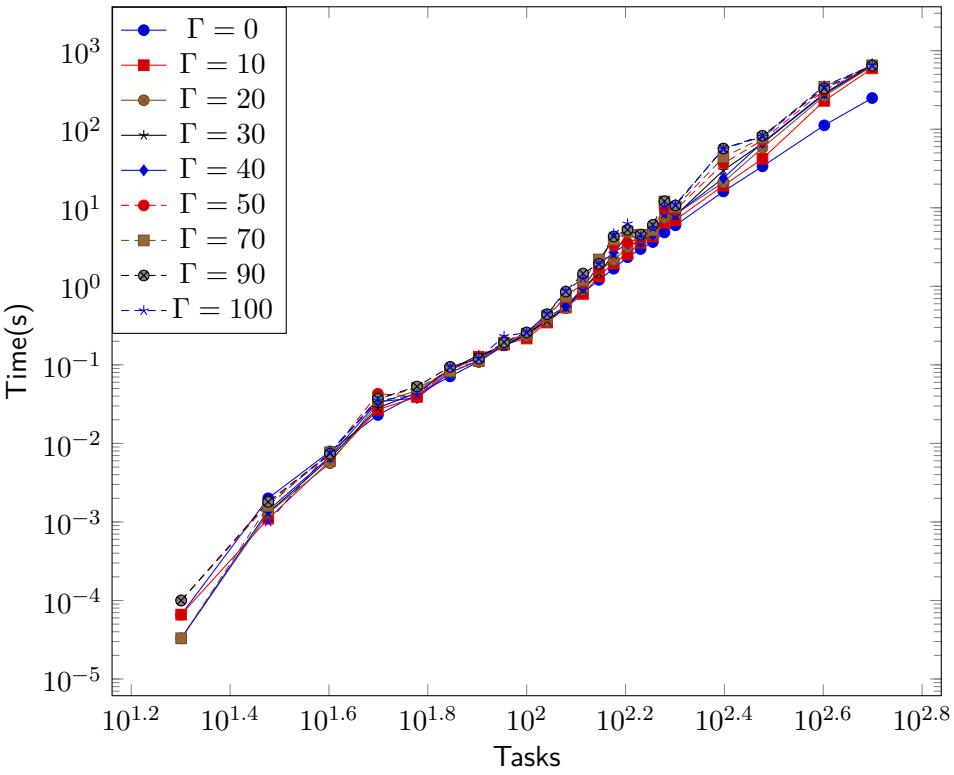


FIGURE 3.2 – Temps d'exécution moyens pour R-HOW avec des budgets d'incertitude différents.

3.5 Expérimentations numériques

Dans cette section nous présentons brièvement les résultats des expérimentations numériques. Une version plus complète de ces expérimentations est disponible dans [Hamaz 2018b] et [Hamaz 2018a].

Nous présentons dans le tableau 3.2 les temps d'exécution moyens de chaque algorithme, l'algorithme de recherche binaire (BS), algorithme itératif (IA) et la version robuste de l'algorithme de Howard (R-HOW), selon le nombre de tâches ainsi que les différentes valeurs du budget d'incertitude.

Le tableau 3.2 montre que la version robuste de l'algorithme de Howard est plus performante que l'algorithme itératif et l'algorithme de recherche binaire. Les temps moyens d'exécution de la version robuste de l'algorithme de Howard sont au moins 100 fois plus petits pour des grandes instances. Notons que dans ce tableau, nous n'avons reporté que les résultats concernant des instances ayant 60 et 70 tâches génériques. De plus, et contrairement à l'algorithme itératif et l'algorithme de recherche binaire, la version robuste de l'algorithme de Howard à résolu toutes les instances (de 10 à 200 tâches génériques) avant d'atteindre le temps limite. Cependant, ce n'est plus le cas quand nous augmentons la valeur du budget d'incertitude.

La figure 3.2 montre, pour chaque valeur du budget d'incertitude variant de 0%

à 100%, le temps d'exécution moyen de la version robuste de l'algorithme de Howard par rapport au nombre de tâches génériques. La figure montre que la version robuste de l'algorithme de Howard n'est pas sensible à la variation du budget d'incertitude, puisque les temps d'exécution moyens restent à peu près stable avec l'augmentation de la valeur du budget d'incertitude. Notons que ce n'est pas le cas pour les deux autres algorithmes car les temps moyens d'exécution associés augmentent significativement avec l'augmentation du budget d'incertitude.

# Tâches	$\Gamma(\%)$	BS (s)	IA (s)	R-HOW	Dev_α (%)
60	0	11.481	0.966	0.042	0.0
	10	86.571	11.447	0.039	7.1
	20	164.886	21.914	0.047	14.3
	30	252.703	33.383	0.044	21.2
	40	343.071	45.071	0.038	26.7
	50	439.351	57.907	0.040	31.0
	70	649.219	85.092	0.051	36.2
	90	836.001	116.024	0.053	37.7
	100	905.110	135.046	0.042	37.7
70	0	20.862	1.719	0.071	0.0
	10	178.332	23.617	0.085	7.3
	20	348.374	45.685	0.089	14.5
	30	528.108	69.333	0.088	21.5
	40	712.859	94.030	0.080	27.1
	50	851.806	122.995	0.091	31.5
	70	970.699	178.415	0.082	36.1
	90	-	266.011	0.094	37.5
	100	-	304.677	0.094	37.5

TABLE 3.2 – Temps d'exécution moyens en secondes pour BS, NCC and R-HOW et les valeurs de pourcentage de déviation des temps de cycle optimaux par rapport aux valeurs nominales des temps de cycle.

3.6 Conclusion

Dans ce chapitre, la version robuste du problème d'ordonnancement cyclique de base a été formalisée. Ensuite, nous avons défini un problème de séparation qui, pour une valeur donnée du temps de cycle, détermine s'il existe un ordonnancement respectant les contraintes de précédence quelque soit le scénario. Enfin, en utilisant ce problème de séparation, nous avons développé deux algorithmes ainsi qu'une extension de l'algorithme de Howard. Les résultats numériques montrent la supériorité de l'algorithme de Howard sur les autres algorithmes malgré sa complexité pseudo-polynomiale. De plus, cet algorithme semble insensible à la variation de la valeur du budget d'incertitude.

Chapitre 4

Problème du job shop cyclique robuste

Sommaire

4.1 Introduction **47**
4.2 Définition du problème **48**
4.3 Approches de résolution pour le $\mathcal{U}^{\Gamma}$-CJSP **50**
4.3.1 Algorithme de Branch-and-Bound pour le $\mathcal{U}^{\Gamma}$-CJSP 51
4.3.2 Méthodes de décomposition pour $\mathcal{U}^{\Gamma}$-CJSP 53
4.4 Expérimentations numériques **56**
4.5 Conclusion **57**

4.1 Introduction

Le problème du job shop cyclique, présenté dans le chapitre 1, est un problème très étudié dans la littérature de l'ordonnancement cyclique. Dans ce chapitre, nous présentons nos contributions dans le cadre de la version robuste de ce problème. Plus précisément, les durées des tâches génériques sont supposées incertaines et sont modélisées par le budget d'incertitude de Bertsimas et Sim [Bertsimas 2004] présenté dans le chapitre 2. Nous considérons un problème bi-niveaux, où les variables sont séparées en deux catégories. Le temps de cycle et les décalages d'occurrence sont les variables de premier niveau et doivent être décidés avant la révélation de l'incertitude. Les variables du second niveau sont représentées par les variables des dates de début des tâches génériques. Celles-ci peuvent être retardées jusqu'à ce que les vraies valeurs des durées des tâches soient connues afin de les prendre en compte dans le processus de décision. L'objectif du problème, que nous appelons $\mathcal{U}^{\Gamma}$-CJSP, est de trouver la plus petite valeur du temps de cycle pour laquelle, pour tout scénario appartenant à l'ensemble d'incertitude, il existe un vecteur de dates de début réalisable.

Le chapitre est structuré comme suit : dans la section 4.2, après avoir fait un rappel du CJSP et présenté l'ensemble d'incertitude, nous proposons deux formulations mathématiques du problème dans sa version robuste. Le première modèle concerne le CJSP robuste dans le cadre statique. Nous montrons que la contrepartie robuste correspond au CSJP déterministe où les tâches génériques prennent leurs valeurs maximales. C'est-à-dire que, dans ce modèle, toutes les tâches génériques dévient de leurs

valeurs nominales. Le deuxième formulation concerne le CJSP robuste dans le cadre bi-niveaux. Dans la section 4.3, nous présentons d'abord le schéma de séparation et d'évaluation (Branch-and-Bound). Ensuite, nous proposons un algorithme de Branch-and-Bound pour le $\mathcal{U}^{\Gamma}$-CJSP, et enfin, nous adaptons deux méthodes de décomposition de problème robuste bi-niveaux.

4.2 Définition du problème

Dans ce chapitre, nous nous intéressons à la version robuste du problème défini dans le paragraphe 1.3 . En effet, chaque tâche générique $i \in \mathcal{T}$ possède une durée p_i qui est généralement considérée comme connue exactement, mais en pratique, des aléas peuvent perturber cette durée en l'augmentant, ce qui peut rendre infaisable une solution déterminée à l'avance. Afin de prendre en compte ce type d'incertitudes, nous proposons d'utiliser un ensemble d'incertitude identique à la section 3.2. L'objectif est de trouver l'ordre des tâches génériques exécutées sur les mêmes machines donnant la plus petite valeur du temps de cycle dans le pire cas, compte tenu des incertitudes sur la durée des tâches.

Pour les mêmes raisons que celles évoquées lors du paragraphe 3.2, il n'est pas intéressant de considérer un modèle statique pour ce problème (excès de conservatisme). Considérons maintenant le cas bi-niveaux. Dans ce contexte, nous allons autoriser les variables représentant les dates de début des occurrences des tâches à être ajustées, et prendre ainsi en compte les retards d'exécution des tâches. Plus précisément, nous allons considérer deux types de variables : les variables du premier niveau sont la variable de temps de cycle α et les variables de décalage d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$, c'est à dire les variables déterminant l'ordre des tâches génériques qui s'exécutent sur une même machine ; les variables du second niveau sont représentées par les dates de début des tâches génériques. Une fois l'aléa observé, les valeurs des dates de début des tâches génériques (variables de second niveau) sont ajustées, afin de respecter les contraintes de précédence et de ressource. Le CJSP robuste bi-niveaux peut être exprimé comme suit :

$$\min \quad \alpha \tag{4.1}$$

$$s.t. \quad \forall \xi \in \mathcal{U}^{\Gamma} : \exists t \geq 0 \begin{cases} t_j - t_i + \alpha H_{ij} \geq p_i(\xi) & \forall (i,j) \in \mathcal{P} \\ t_j - t_i + \alpha K_{ij} \geq p_i(\xi) & \forall (i,j) \in \mathcal{D} \end{cases} \tag{4.2}$$

$$K_{ij} + K_{ji} = 1 \qquad \forall (i,j) \in \mathcal{D} \tag{4.3}$$

$$K_{ij} \in \mathbb{Z} \qquad \forall (i,j) \in \mathcal{D} \tag{4.4}$$

$$\alpha \geq 0 \tag{4.5}$$

Nous cherchons ici un temps de cycle, un vecteur de variables de décalage d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$ et un vecteur $(t_i)_{i\in\mathcal{T}}$ des dates de début des tâches génériques pour chaque scénario possible $\xi \in \mathcal{U}^{\Gamma}$.

Nous remarquons que la formulation (4.1)-(4.5) peut difficilement être directement

résolu par des outils de programmation mathématique à cause du très grand nombre de contraintes (4.2) qui doivent être satisfaites pour chaque valeur de $\xi \in \mathcal{U}^{\Gamma}$. Ce type de problème est généralement résolu par des méthodes de décomposition ou approximé par des règles de décision.

Job	$\mathcal{J}_1$				$\mathcal{J}_2$			
Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Duration	[3,4]	[2,4]	[3,4]	[2,3]	[2,5]	[1,2]	[3,4]	[2,8]
Machine	M_1	M_1	M_3	M_1	M_2	M_1	M_3	M_2

TABLE 4.1 – Données de l'exemple 8.

Exemple 8. *Considérons un exemple de jobshop tel que $\mathcal{T} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ de 8 tâches, 2 jobs, 3 machines et $WIP = 2$. Les données du tableau 4.1 décrivent l'affectation des tâches aux machines et leur durée qui est décrite par un intervalle. Le graphe de précédence associé à ce problème est donné par la figure 4.1 (pour des raisons de clarté, les arcs disjonctifs ne sont pas représentés).*

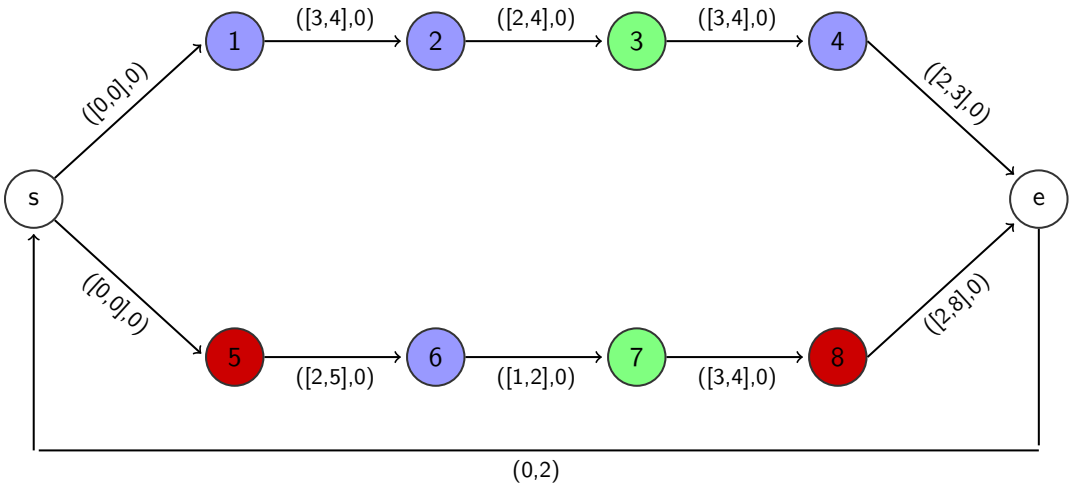


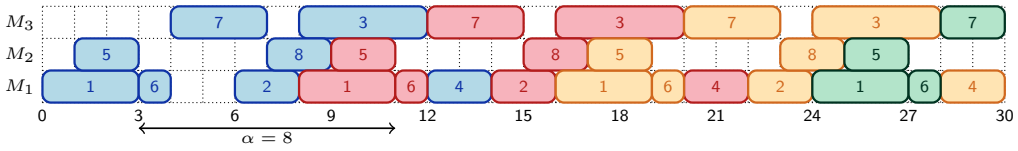
FIGURE 4.1 – Graphe de précédence associé à l'exemple 8.

Considérons trois ordonnancements différents s_1, s_2 et s_3 décrits par le tableau 4.2. Les trois ordonnancements conduisent à un temps de cycle de 8 u.t. dans des conditions nominales (pas de déviation des temps de traitement) et sont représentés sur la figure 4.2. Cependant, ces ordonnancements ne se comportent pas de la même manière lorsque des incertitudes surviennent. Plus précisément, en considérant $\Gamma = 1$, l'ordonnancement s_1 conduit à un temps de cycle de 13 u.t. lorsque $\xi_8 = 1$. Dans les mêmes conditions, le programme s_3 conduit à un temps de cycle de 12 u.t. alors que le programme s_2 a un temps de cycle de 11 u.t. pour son écart le plus défavorable ($\xi_3 = 1$). De plus, l'augmentation du budget d'incertitude à $\Gamma = 2$ conduit à un temps de cycle de 17 u.t. (resp. 14 u.t. et 14 u.t.) pour l'ordonnancement s_1 (resp. s_2 et s_3).

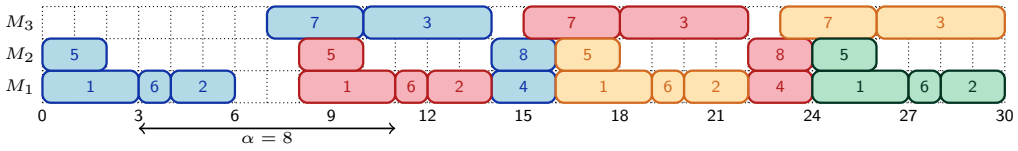
Parmi ces 3 ordonnancements, nous pouvons conclure que s_2 est le plus robuste lorsque l'on considère $\Gamma = 1$ et que s_2 et s_3 sont les plus robustes lorsque $\Gamma = 2$ (mais avec différents scénarios de pire cas). Tous ces résultats sont résumés dans les dernières colonnes du tableau 4.2 et dans les figures 4.3 et 4.4.

	K_{12}	K_{14}	K_{16}	K_{24}	K_{26}	K_{46}	K_{58}	K_{37}	$\alpha_{\Gamma=0}$	$\alpha_{\Gamma=1}$	$\alpha_{\Gamma=2}$
s_1	0	-1	0	0	1	2	0	1	8	13	17
s_2	0	-1	0	-1	1	2	-1	1	8	11	14
s_3	0	-1	0	0	0	1	-1	0	8	12	14

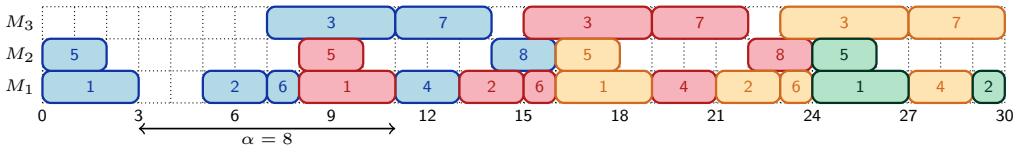
TABLE 4.2 – Trois ordonnancement de l'exemple 8 et leur pire temps de cycle pour différentes valeurs de Γ.



(a)



(b)



(c)

FIGURE 4.2 – Pour $\Gamma = 0$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3.

4.3 Approches de résolution pour le $\mathcal{U}^\Gamma$-CJSP

Cette section est consacrée à la résolution du $\mathcal{U}^\Gamma$-CJSP. Nous avons développé trois méthodes de résolution pour ce problème. La première méthode est un algorithme de Branch-and-Bound , les deux autres sont des méthodes de résolution classiques pour les problèmes robustes bi-niveaux, que nous adaptons au problème considéré. Plus précisément, la première méthode de décomposition est un algorithme de génération différée de colonnes et de contraintes, et la deuxième est une méthode de génération de contraintes.

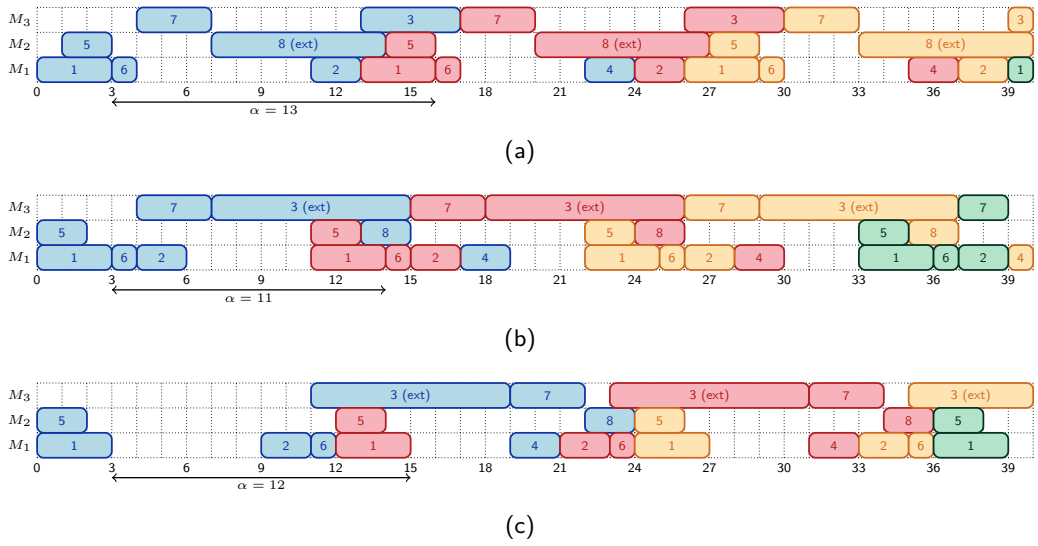


FIGURE 4.3 – Pour $\Gamma = 1$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3.

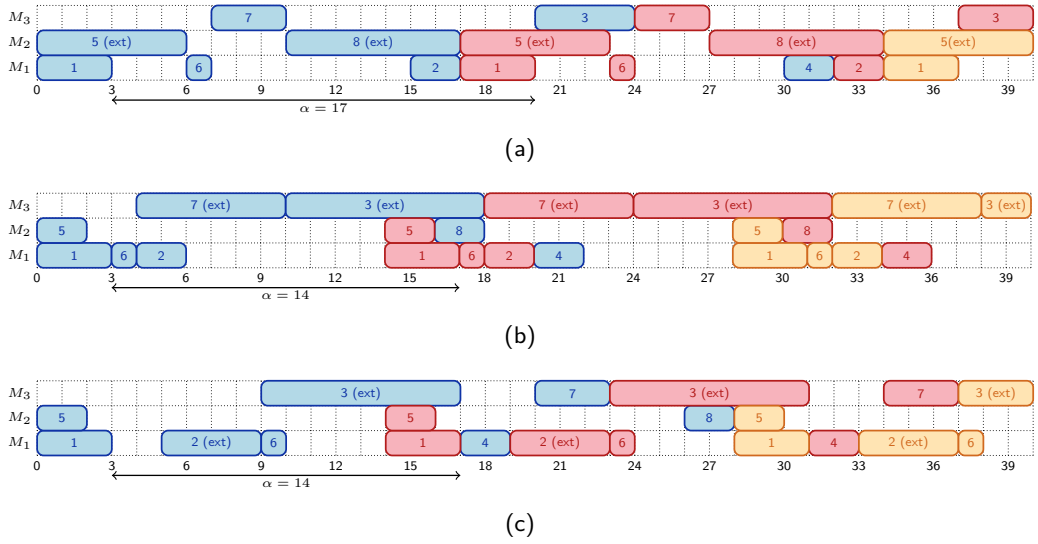


FIGURE 4.4 – Pour $\Gamma = 2$, (a) Ordonnancement s_1. (b) Ordonnancement s_2. (c) Ordonnancement s_3.

Dans ce qui suit, en exploitant les résultats obtenus pour le $\mathcal{U}^{\Gamma}$-BCSP, nous allons décrire un algorithme de Branch-and-Bound permettant de résoudre le $\mathcal{U}^{\Gamma}$-CJSP.

4.3.1 Algorithme de Branch-and-Bound pour le $\mathcal{U}^{\Gamma}$-CJSP

Dans cette sous-section, nous allons adapter le schéma de séparation et d'évaluation au $\mathcal{U}^{\Gamma}$-CJSP. A cet effet, nous allons présenter quelques choix spécifiques à notre problème, tel le choix de l'ensemble des solutions à explorer, la règle de branchement,

le calcul des bornes, *etc*.

L'algorithme de Branch-and-Bound que nous présentons se base sur la remarque suivante :

Remarque 2. *Étant donné un vecteur $(\bar{K}_{ij})_{(i,j)\in\mathcal{D}}$ de décalage d'occurrence, c'est-à-dire un ordre d'exécution des tâches génériques sur les mêmes machines, trouver la plus petite valeur du temps de cycle α de sorte que, pour tout scénario $\xi \in \mathcal{U}^\Gamma$, il existe un ordonnancement, revient à résoudre le problème suivant :*

$$\min \quad \alpha \tag{4.6}$$

$$s.t. \quad \forall \xi \in \mathcal{U}^\Gamma : \exists t \geq 0 \begin{cases} t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq p_i(\xi) & \forall (i,j) \in \mathcal{P} \\ t_j(\xi) - t_i(\xi) + \alpha \bar{K}_{ij} \geq p_i(\xi) & \forall (i,j) \in \mathcal{D} \end{cases} \tag{4.7}$$

$$\alpha \geq 0 \tag{4.8}$$

Nous remarquons que ce problème correspond au $\mathcal{U}^\Gamma$-BCSP. Donc, lors de l'application d'un schéma de Branch-and-Bound, nous pouvons résoudre chacun des sous-problèmes par un des algorithmes que nous avons développé dans le chapitre 3.

L'idée générale de la méthode du Branch-and-Bound pour le $\mathcal{U}^\Gamma$-CJSP est d'explorer l'espace des valeurs réalisables des décalages d'occurrence. Les branchements, sur la base de la remarque 2, vont donner lieu à des sous-problèmes qui correspondent à des $\mathcal{U}^\Gamma$-BCSP. Par conséquent, les algorithmes décrits dans le chapitre 3 peuvent être utilisés pour les résoudre.

Chaque noeud du Branch-and-Bound correspond à un sous problème défini par le sous-graphe $G_s = (\mathcal{T}, E \cup \mathcal{D}_s)$, où $\mathcal{D}_s \subseteq \mathcal{D}$ est le sous-ensemble de décalages d'occurrences déjà fixés. L'algorithme commence avec un noeud racine de manière à ce qu'il n'y ait pas de décalage d'occurrence déjà fixé (le sous-graphe dans ce cas est G_{root}, où $\mathcal{D}_{root} = \emptyset$). Ensuite, le branchement est effectué en fixant un décalage d'occurrence $K_{ij} \in \mathcal{D}$ qui n'est pas encore déterminé. Ceci engendre un noeud fils pour chaque valeur de K_{ij}. La variable de décalage d'occurrence de K_{ij} peut prendre des valeurs dans $\mathbb{Z}$, mais il existe une méthode afin de calculer une borne inférieure K_{ij}^- et une borne supérieure $1 - K_{ji}^-$ sur les valeurs qui sont réalisables (voir Chapitre 1). Ces noeuds fils sont ensuite évalués en calculant la valeur du temps de cycle α dans le pire cas des réalisations des durées des tâches dans l'ensemble de scénarios $\mathcal{U}^\Gamma$. Cette évaluation est effectuée en utilisant l'algorithme de Howard adapté qui a été présenté dans le chapitre 2. Ce choix est fait sur la base des expérimentations numériques qui ont été conduites. Cet algorithme explore l'arbre de recherche en utilisant la stratégie du meilleur d'abord (Best First Strategy : BeFS), c'est-à-dire qu'à chaque branchement, nous choisissons le noeud ayant la meilleure borne inférieure. Le choix de cette stratégie est dû au fait qu'elle est susceptible d'engendrer de bonnes solutions faisables. Notons qu'une solution faisable est atteinte quand tous les décalages d'occurrence sont fixés.

Dans ce qui suit, nous allons décrire plus en détail les différentes étapes de l'algorithme de Branch-and-Bound .

Schéma de branchement et règle de branchement

Comme déjà rappelé dans le chapitre 1, à notre connaissance, seulement deux schémas de branchement ont été présentés dans la littérature pour le CJSP. Dans ces deux schémas de branchement, les branchements sont effectués sur les décalages d'occurrence qui ne sont pas encore fixés. Le premier schéma de branchement à été introduit dans [Hanen 1994] et est basé sur l'intervalle des valeurs possibles pour les décalages d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$. L'auteur utilise un branchement dichotomique. Plus précisément, à partir d'un noeud de l'arbre de recherche, le branchement est effectué sur un arc disjonctif K_{ij} et deux noeuds sont générés. Dans le premier noeud, l'intervalle des valeurs possibles pour K_{ij} est restreint à $I_{ij} = [K_{ij}^{-}, c_{ij}]$ et dans le deuxième à $[c_{ij}+1, 1-K_{ji}^{-}]$, où c_{ij} est le milieu de l'intervalle I_{ij}. Le second schéma de branchement est introduit dans [Fink 2012]. Le branchement consiste à identifier une variable de décalage d'occurrence $K_{ij} \in \mathcal{D}$ qui n'est pas encore fixée et générer un noeud fils pour chaque valeur possible dans l'intervalle I_{ij}. Pour chaque noeud généré, l'algorithme affecte une valeur différente dans I_{ij} pour K_{ij}.

En ce qui concerne notre algorithme de Branch-and-Bound, nous utilisons le même schéma de branchement que celui introduit dans [Fink 2012]. Ce choix de schéma de branchement est motivé par le fait que, dans chaque noeud du Branch-and-Bound, le sous-problème généré correspond à un $\mathcal{U}^{\Gamma}$-BCSP, donc nous pouvons utiliser les algorithmes développés dans le chapitre 2 afin de résoudre chacun de ces sous-problèmes. Ceci nous permettra d'avoir une borne inférieure sur la valeur du temps de cycle α garantissant, pour chaque $\xi \in \mathcal{U}^{\Gamma}$, l'existence d'un ordonnancement périodique $\sigma = (t, \alpha_{\xi})$ tel que $\alpha_{\xi} \leq \alpha$. Nous avons testé différentes règles de branchement, et il s'avère que celle selon laquelle nous branchons sur les variables de décalage d'occurrence K_{ij}, où $K_{ij}^{-} + K_{ji}^{-}$ est maximum, donne les meilleurs temps d'exécution. Ceci peut être expliqué par le fait que cette règle de branchement génère un petit nombre de noeuds fils à chaque branchement, ce qui limite la taille de l'arbre de recherche.

4.3.2 Méthodes de décomposition pour $\mathcal{U}^{\Gamma}$-CJSP

Cette section est consacrée à l'adaptation des méthodes de décomposition classiques pour les problèmes d'optimisation robuste bi-niveaux au problème $\mathcal{U}^{\Gamma}$-CJSP. Plus précisément, nous adaptons la méthode de génération différée de contraintes et la méthode de génération différée de colonnes et de contraintes [Ayoub 2016]. Dans notre cas, les variables de premier niveau sont constituées du temps de cycle et des variables de décalage des occurrences. Les dates de début des tâches correspondent alors aux variable de recours de notre problème.

La proposition suivante caractérise le problème de séparation d'une solution.

Proposition 2. *Soit $\alpha^* \in \mathbb{R}+$ et $(K_{ij}^*)_{(i,j)\in\mathcal{D}} \in \mathbb{Z}^{|\mathcal{D}|}$ respectivement un temps de cycle fixé et un vecteur de décalage fixé. Alors, le temps de cycle α^* est réalisable pour chaque scénario $\xi \in \mathcal{U}^{\Gamma}$ si et seulement si la valeur du programme linéaire en variables mixtes*

suivant :

$$\max \quad \sum_{e\in\mathcal{P}}(\bar{p}_e - H_e\alpha^*)u_e + \sum_{e\in\mathcal{D}}(\bar{p}_e - K^*_e\alpha^*)u_e + \sum_{e\in\mathcal{P}\cup\mathcal{D}}\hat{p}_e s_e \tag{4.9}$$

$$s.t. \quad \sum_{i\in\mathcal{T}}\xi_i \leq \Gamma \tag{4.10}$$

$$\sum_{e\in\sigma^-(i)} u_e - \sum_{e\in\sigma^+(i)} u_e = 0 \qquad \forall\, i \in \mathcal{T} \tag{4.11}$$

$$s_e \leq u_e \qquad \forall\, e \in \mathcal{P} \tag{4.12}$$

$$s_e \leq \xi_e \qquad \forall\, e \in \mathcal{P} \tag{4.13}$$

$$1 \geq u_e \geq 0 \qquad \forall\, e \in \mathcal{P} \tag{4.14}$$

$$1 \geq s_e \geq 0 \qquad \forall\, e \in \mathcal{P} \tag{4.15}$$

$$\xi_i \in \{0,1\} \qquad \forall\, i \in \mathcal{T} \tag{4.16}$$

possède une solution optimale non-positive.

Preuve. Une fois les variables de décalage d'occurrence fixées, le problème correspond au $\mathcal{U}^\Gamma$-BCSP. Le problème de séparation est formulé de la même manière que pour le $\mathcal{U}^\Gamma$-BCSP dans le chapitre 3 (voir la proposition 1). □

Les deux algorithmes que nous proposons se basent sur ce problème de séparation. Le principe de ces algorithmes est de résoudre un problème dit *problème maître* qui contient uniquement un sous-ensemble des contraintes $\mathcal{S} \subset \mathcal{U}^\Gamma$. Ensuite, un sous-problème fournit un scénario $\xi \in \mathcal{U}^\Gamma$ pour lequel la solution $(\alpha^*, (K^*_{ij})_{(i,j)\in\mathcal{D}})$ n'est pas réalisable, et un vecteur dual $(u_e)_{e\in\mathcal{P}}$. Ce qui différencie les deux algorithmes est la façon d'incorporer cette information dans le problème maître afin d'éliminer cette solution de l'espace de recherche. Le schéma général associé à cet algorithme est donné dans la figure 4.5.

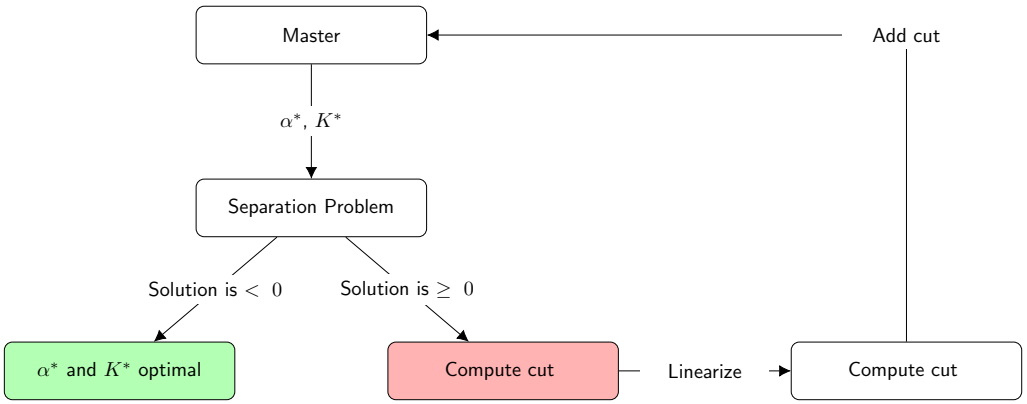


FIGURE 4.5 – Schéma général de l'algorithme de décomposition.

Le problème maître est formulé de la façon suivante :

$$\min \quad \alpha \tag{4.17}$$

$$s.t. \quad t_j(\xi) - t_i(\xi) + \alpha H_{ij} \geq \bar{p}_i + \hat{p}_i \xi \quad \forall (i,j) \in \mathcal{P}, \xi \in \mathcal{S} \tag{4.18}$$

$$t_j(\xi) - t_i(\xi) + \alpha K_{ij} \geq \bar{p}_i + \hat{p}_i \xi \quad \forall (i,j) \in \mathcal{D}, \xi \in \mathcal{S} \tag{4.19}$$

$$K_{ij} + K_{ji} = 1 \quad \forall (i,j) \in \mathcal{D} \tag{4.20}$$

$$K_{ij}^{-} \leq K_{ij} \leq 1 - K_{ji}^{-} \quad \forall (i,j) \in \mathcal{D} \tag{4.21}$$

$$K_{ij} \in \mathbb{Z} \quad \forall (i,j) \in \mathcal{D} \tag{4.22}$$

De plus, le MIP (4.17)-(4.22) peut être linéarisé de la même façon que dans §1.3.

Méthode de génération différée de contraintes

La méthode de génération différée de contraintes est similaire à la décomposition de Benders. Rappelons que le principe de la décomposition de Benders est de résoudre un problème restreint (problème master), et ensuite, en résolvant un problème de séparation, générer soit une coupe de faisabilité soit une coupe d'optimalité à ajouter au problème restreint. De manière similaire, notre objectif est de trouver une coupe qui, une fois ajoutée au problème restreint, va interdire la solution courante $(\alpha^*, (K_{ij}^*)_{(i,j)\in\mathcal{D}})$. Ceci peut être fait par le biais de la coupe de Benders classique suivante :

$$\sum_{e\in\mathcal{P}} (\bar{p}_e + \hat{p}_e \xi_e^* - H_e \alpha) u_e^* + \sum_{e\in\mathcal{D}} (\bar{p}_e + \hat{p}_e \xi_e^* - K_e \alpha) u_e^* \leq 0. \tag{4.23}$$

Cela revient à forcer la valeur de l'objectif du problème de séparation à être non-postive. Ce qui interdira la solution de premier niveau actuelle. Nous remarquons que cette contrainte n'est pas linéaire, dû à la multiplication des variables de décalage d'occurrence $(K_{ij})_{(i,j)\in\mathcal{D}}$ avec la variable du temps de cycle α. Nous pouvons linéariser ce produit en introduisant la variable de taux de production $\tau = \frac{1}{\alpha}$, ce qui donne la contrainte suivante :

$$\sum_{e\in\mathcal{P}} \left(\tau \left(\bar{p}_e + \hat{p}_e \xi_e^*\right) - H_e\right) u_e^* + \sum_{e\in\mathcal{D}} \left(\tau \left(\bar{p}_e + \hat{p}_e \xi_e^*\right) - K_e\right) u_e^* \leq 0. \tag{4.24}$$

Nous avons donc deux formulations des contraintes générées.Une fois le problème de séparation résolu, la coupe (4.23) mettant en jeu la variable α est obtenue. Afin de la linéariser pour l'incorporer au problème restreint, nous revenons à une formulation avec la variable τ (la coupe (4.24)).

Méthode de génération différée de colonnes et de contraintes

La méthode de génération différée de colonnes et de contraintes fonctionne d'une manière similaire à la méthode précédente. Le problème restreint, puis le problème de séparation sont d'abord résolus. Deux cas se présentent : soit la valeur de l'objectif est

non-positive, et dans ce cas la solution courante $(\alpha^*, (K^*_{ij})_{(i,j)\in\mathcal{D}})$ est optimale, soit la solution est non faisable. Nous pouvons récupérer le scénario $\xi^* \in \mathcal{U}^\Gamma$ pour lequel la solution courante n'est pas faisable et le vecteur de la solution duale $(u^*_e)_{e\in E\cup\mathcal{D}}$. La différence entre cette méthode et la précédente consiste en la manière avec laquelle cette information est remontée au problème restreint. En effet, au lieu d'ajouter la coupe de Benders au problème restreint, cette méthode ajoute un bloc de variables et de contraintes correspondant au scénario $\xi^* \in \mathcal{U}^\Gamma$, qui peut se formuler comme suit :

$$t_j(\xi^*_i) - t_i(\xi^*_i) + \alpha H_{ij} \geq p_i(\xi^*_i) \quad \forall\,(i,j) \in E \tag{4.25}$$
$$t_j(\xi^*_i) - t_i(\xi^*_i) + \alpha K_{ij} \geq p_i(\xi^*_i) \quad \forall\, s \in \mathcal{M}, \forall\, i,j \in \mathcal{T}_s. \tag{4.26}$$

Ce bloc de variables et de contraintes est linéarisé en posant $\tau = \frac{1}{\alpha}$, ce qui donne :

$$u_j(\xi^*_i) - u_i(\xi^*_i) + H_{ij} \geq \tau p_i(\xi^*_i) \quad \forall\,(i,j) \in E \tag{4.27}$$
$$u_j(\xi^*_i) - u_i(\xi^*_i) + K_{ij} \geq \tau p_i(\xi^*_i) \quad \forall\, s \in \mathcal{M}, \forall\, i,j \in \mathcal{T}_s. \tag{4.28}$$

4.4 Expérimentations numériques

Les expérimentation numériques sont détaillées dans [Hamaz 2022] mais un bref résumé est exposé dans ce paragraphe.

Le tableau 4.3 présente les temps moyens d'exécutions des instances ayant de 30 à 40 tâches avec des budgets d'incertitude allant de 0% to 100%. Toutes les instances ont été résolues d'une manière optimale par l'algorithme de Branch-and-bound avant d'atteindre les limites de temps. Pour cette méthode, les temps d'exécutions moyens montrent des valeurs peu sensibles par rapport à la variation du budget d'incertitude.

Concernant les deux algorithmes de décomposition, toutes les instances ayant 10 et 20 tâches ont été résolues à l'optimum (non détaillé ici), ce qui n'est pas le cas pour les instances ayant 30 et 40 tâches. Nous observons que les temps moyens d'exécution des méthodes de décomposition sont en général plus grands que ceux de l'algorithme de Branch-and-Bound. De plus, ceux associés à l'algorithme de génération différée de colonnes et de contraintes sont sensiblement meilleurs que ceux de l'algorithme de génération différée de contraintes.

Ce tableau montre aussi le pourcentage de déviation, pour un certain budget d'incertitude Γ, du temps de cycle optimal α_Γ du temps de cycle nominal α_{nom}, où toutes les tâches prennent leurs durées nominales. Ce pourcentage est calculé comme par $\mathrm{Dev}_\alpha = \frac{\alpha_\Gamma - \alpha_{nom}}{\alpha_{nom}}$. En d'autres termes, ce pourcentage représente la valeur du temps de cycle à augmenter afin de protéger l'ordonnancement des incertitudes. Cette valeur est communément nommée "coût de la robustesse". Nous remarquons que ce pourcentage de déviation se stabilise quand le budget d'incertitude est supérieur à 20 ou 30 pourcent. Ceci peut arriver quand le nombre de noeuds du circuit ayant le plus d'arcs est inférieur à Γ. Dans ce cas, augmenter la valeur de Γ n'influe plus sur la valeur du temps de cycle optimal.

Le tableau 4.4 montre le nombre d'instances résolues avant d'atteindre le temps

limite. Ces résultats concernent les instances ayant de 50 à 100 tâches. Le tableau montre que les trois algorithmes n'arrivent pas à résoudre toutes les instances en moins de 900 seconds. Par exemple, pour l'algorithme de Branch-and-Bound et pour les instances ayant 100 tâches, 10 jobs et 20 machines, seulement 6 instances ont été résolues dans temps imparti. Cependant, l'algorithme de Branch-and-Bound reste le plus efficace parmi les trois algorithmes.

					B&B		RG		RCG	
n	$\|j\|$	$\|m\|$	$\Gamma(\%)$	$\mathrm{Dev}_\alpha(\%)$	T (s)	nb_ins	T (s)	nb_ins	T (s)	nb_ins
30	5	8	0	0	22.64	20	1.99	18	1.82	18
			10	38.25	12.00	20	64.38	17	122.04	17
			20	49.03	15.40	20	90.91	17	117.63	18
			30	50.91	66.24	20	240.92	16	54.92	17
			40	50.91	16.30	20	254.99	11	44.88	17
			50	50.91	17.11	20	183.67	9	38.63	17
			70	50.91	13.83	20	274.77	11	7.74	17
			90	50.91	15.75	20	267.00	12	10.69	17
			100	50.91	15.82	20	266.17	12	20.80	17
40	4	8	0	0	138.91	20	4.87	17	24.83	17
			10	37.92	55.92	20	108.48	13	513.05	10
			20	54.46	92.14	20	339.54	8	368.12	8
			30	54.91	96.75	20	289.09	3	363.92	12
			40	54.91	134.44	20	630.48	16	203.74	13
			50	54.91	155.23	20	-	0	80.39	15
			70	54.91	187.20	20	-	0	41.45	15
			90	54.91	204.48	20	673.78	1	68.92	16
			100	54.91	177.24	20	663.18	1	85.24	16

TABLE 4.3 – Temps d'exécution moyens en secondes, pourcentage de la valeur de déviation du temps de cycle optimal par rapport au temps de cycle nominal ainsi que le nombre d'instances résolues parmi 20 pour l'algorithme de Branch-and-Bound (B&B), l'algorithme de génération différée de contraintes (RG) et l'algorithme de génération de colonnes et de contraintes (RCG).

La figure 4.6 indique le pourcentage d'instances résolues pour plusieurs valeurs du budget d'incertitude avec un jeu de 20 instances et un temps limité à 900 secondes. Ces résultats corroborent les remarques précédentes sur l'efficacité l'algorithme de Branch-and-Bound sur ce problème.

4.5 Conclusion

Dans ce chapitre, nous avons étudié le problème du job shop cyclique où les durées des tâches sont incertaines et modélisées par un ensemble d'incertitude de type budget. Le but est de trouver le meilleur ordre des tâches génériques exécutées sur les mêmes machines, de sorte que, quelles que soit les réalisations des durées des tâches, le temps de cycle est minimum. Dans un premier temps, nous avons proposé un algorithme de Branch-and-Bound où nous avons pu exploiter les méthodes de résolution développées dans le chapitre précèdent pour le problème d'ordonnancement cyclique de base. Ensuite, nous avons proposé deux approches de décomposition classiques pour les

				B&B	RG	RCG
$\|t\|$	$\|j\|$	$\|m\|$	$\Gamma(\%)$	nb_ins	nb_ins	nb_ins
			0	11	16	15
			10	10	3	0
			20	11	2	1
			30	14	1	2
50	5	10	40	13	0	3
			50	13	0	5
			70	12	1	0
			90	12	1	7
			100	12	1	7
			0	7	4	3
			10	5	0	1
			20	4	0	1
			30	4	0	2
60	6	10	40	4	0	2
			50	3	0	2
			70	3	0	3
			90	1	0	3
			100	1	0	3
			0	8	0	0
			10	8	0	0
			20	4	0	0
			30	2	0	0
80	8	16	40	2	0	0
			50	2	0	0
			70	2	0	0
			90	0	0	0
			100	0	0	0
			0	0	0	0
			10	0	0	0
			20	3	0	0
			30	2	0	0
100	10	20	40	1	0	0
			50	0	0	0
			70	0	0	0
			90	0	0	0
			100	0	0	0

TABLE 4.4 – Nombre d'instances résolues en moins de 900 secondes parmi 20 pour l'algorithme de Branch-and-Bound (B&B), l'algorithme de génération différée de contraintes (RG) et l'algorithme de génération de colonnes et de contraintes (RCG)

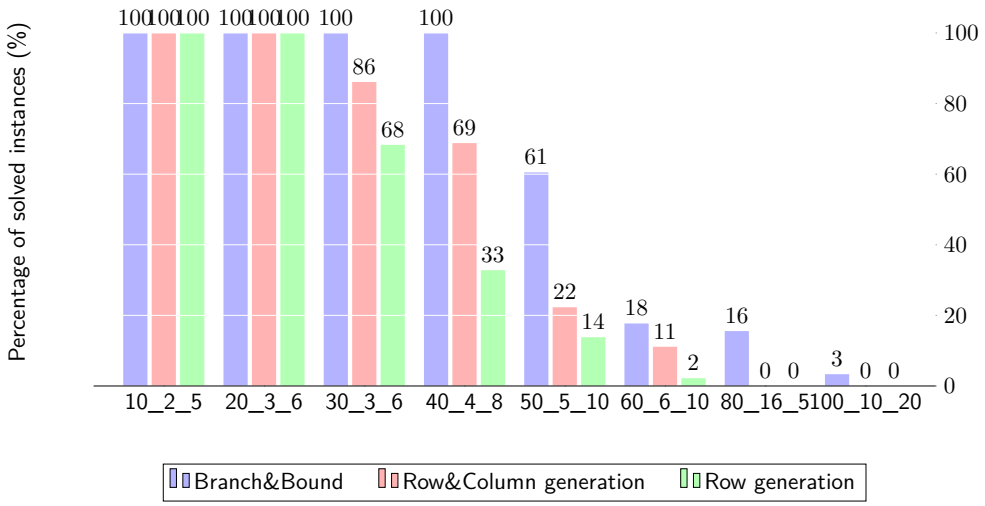


FIGURE 4.6 – Nombre d'instances résolues pour les différentes approches du $\mathcal{U}^{\Gamma}$-CJSP.

problèmes d'optimisation robuste bi-niveaux. Afin de montrer l'efficacité de nos algorithmes, nous avons conduit des expérimentations numériques. Ces dernières révèlent que l'algorithme de Branch-and-Bound est nettement plus efficace que les algorithmes de décomposition classiques pour ce problème d'optimisation robuste bi-niveaux.

Chapitre 5

Modèles et algorithmes pour le problème du job shop cyclique flexible

Sommaire

5.1 Introduction 61
5.2 Définition du problème 62
5.3 Approches de résolution 63
5.3.1 Modèle mathématique 63
5.3.2 Méthode de décomposition 64
5.3.3 Méthodes approchées 66
5.4 Expérimentations numériques 67
5.5 Conclusion 67

5.1 Introduction

Dans ce chapitre, nous traitons une extension du problème de job shop cyclique qui a été défini dans la section 1.3. Plus précisément, ce chapitre concerne les problèmes de job shop cyclique flexible, c'est-à-dire que plusieurs machines sont candidates pour effectuer les tâches élémentaires. L'exécution reste non-préemptive et une tâche est exécutée par une seule machine. De nouvelles variables de décision vont donc enrichir le modèle (1.7) puisque l'ordonnancement doit aussi affecter des machines aux tâches.

Ce type de problème est particulièrement présent dans les installations robotisées. Les robots eux-mêmes ont vu leur flexibilité progresser. En effet, la tendance actuelle est à l'utilisation de ces nouveaux robots qui s'opposent à l'ancienne génération dite "fixe" ou "rigide". Ces outils flexibles nécessitent généralement un investissement financier plus important, mais l'économie à long terme est réelle. Par conséquent, les cellules robotisées ont eu un développement important ces derniers années et ont amené avec elles de nouveaux problèmes d'optimisation. En réalité, la flexibilité concerne toute la ligne de production puisque les durées de production sont désormais de plus en plus courtes.

Concernant les problèmes d'ordonnancement sur machines flexibles (telles des cellules robotisées), les travaux d'optimisation sur le problème du job shop flexible (non

périodique), ont débuté aux débuts des années 90 avec [Brucker 1990]. Assez peu de méthodes exactes sont présentes dans la littérature ([Yu 2017] et [Ben Hmida 2010]) qui est principalement composée de solutions à base de recherche tabou ou d'algorithme évolutionnaire (sans être exhaustif [Nouri 2018], [Azzouz 2017], [Chen 1999] et [Jalilvand-Nejad 2013]). Nous pouvons néanmoins noter un intérêt croissant pour ce sujet et une littérature récente qui fait appel aux techniques de programmation mathématique. [Yu 2017] et [Ben Hmida 2010]

En revanche, les recherches sur l'ordonnancement cyclique flexible sont peu nombreuses et plutôt récentes. Parmi les méthodes exactes, nous pouvons mentionner [Zhang 2015] et [Ghadiri Nejad 2018] mais les références concernant les méthodes approchées sont bien plus nombreuses ([Bożejko 2015], [Kechadi 2013] et [Hsu 2002]).

Le travail de ce chapitre a fait l'objet des publications [Quinton 2020] et [Quinton 2018].

5.2 Définition du problème

Le problème du job shop flexible cyclique (FCJSP) est un CJSP (défini à la Section 1.3) dans lequel les machines sont flexibles. En effet, dans ce problème pour chaque tâche $i \in \mathcal{T}$, il existe un sous-ensemble $R(i) \subset \mathcal{R}$ de ressources qui sont candidates pour exécuter la tâche i. Par conséquent, l'affectation de la tâche i à une machine $r \in \mathcal{R}$ est une variable de décision. La conséquence principale de cette flexibilité sur le modèle classique de CJSP est que l'ensemble $\mathcal{D}$ des contraintes de disjonction dépend de l'affectation des tâches aux machines. Le FCJSP est NP-difficile puisque une fois le problème d'affectation des tâches sur les machines résolu, on se ramène à un problème de CJSP qui est lui-même NP-difficile.

Exemple 9. *Le tableau 5.1 représente les données d'un FCJSP composé de 7 tâches élémentaires et 2 jobs. Chaque tâche peut être exécutée sur un sous ensemble de 3 machines parmi 4. Les contraintes de précédence sont formées par deux travaux* $J_1 = O_{11}O_{12}O_{13}O_{14}$ *et* $J_2 = O_{25}O_{26}O_{27}$ *. La figure 5.1 représente une solution possible pour ce problème. Pour l'affectation considérée dans la figure 5.1, le plus petit temps de cycle possible est 13, la ressource* M_1 *étant saturée.*

TABLE 5.1 – Un exemple de FCJSP

Tâche	1			2			3			4		
Machine	M_1	M_2	M_4	M_1	M_2	M_3	M_2	M_3	M_4	M_1	M_2	M_3
Durée	5	6	6	4	5	6	7	5	6	4	7	5
Tâche	5			6			7					
Machine	M_1	M_2	M_3	M_1	M_3	M_4	M_1	M_2	M_3			
Durée	4	4	5	3	4	2	5	7	5			

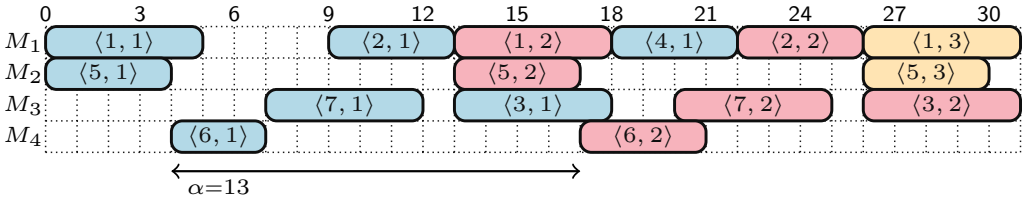


FIGURE 5.1 – Un ordonnancement réalisable de l'exemple 9.

5.3 Approches de résolution

Dans cette partie nous proposons tout d'abord un programme linéaire mixte ainsi que deux heuristiques pour résoudre ce problème combinatoire.

5.3.1 Modèle mathématique

Soit $R(i,j) = R(i) \cap R(j), \forall i,j \in \mathcal{T}$ l'ensemble des machines candidates à l'exécution de la tâche i et de la tâche j.

Afin de déterminer sur quelle ressource est exécutée une tâche i, nous introduisons

$$\forall i \in \mathcal{T}, \forall r \in R(i), \quad m_{ir} = \begin{cases} 1 \text{ si la tâche } i \text{ est exécutée sur la machine } r \\ 0 \text{ sinon.} \end{cases}$$

En partant du modèle (1.7) du problème du CJSP exposé dans le chapitre 1.3, on peut définir le programme mixte suivant pour résoudre le FCJSP :

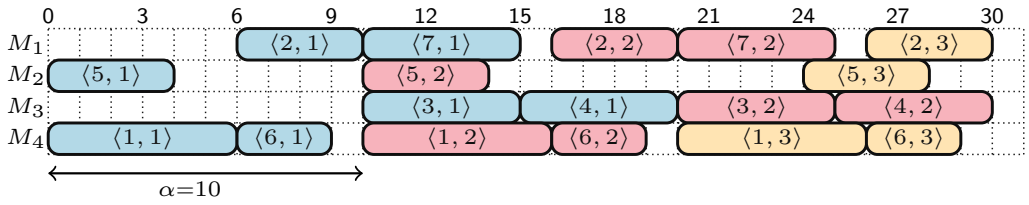


FIGURE 5.2 – Ordonnancement produit par le modèle (5.1).

$$\max \tau$$

s.t.

$$\tau \leq \sum_{r \in R(i)} \frac{m_{ir}}{p_{ir}}, \quad \forall i \in \mathcal{T} \tag{5.1a}$$

$$u_j + H_{ij} \geq u_i + \sum_{r \in R(i)} p_{ir} y_{ir}, \quad \forall (i,j) \in \mathcal{E} \tag{5.1b}$$

$$P_1(2 - m_{ir} - m_{jr}) + u_j + K_{i,j} \geq u_i + p_{ir} y_{ir}, \quad \forall (i,j) \in \mathcal{D}, \forall r \in R(i,j) \tag{5.1c}$$

$$\sum_{r \in R(i)} y_{ir} = \tau, \quad \forall i \in \mathcal{T} \tag{5.1d}$$

$$y_{ir} \leq m_{i,r}, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{5.1e}$$

$$y_{ir} \geq 0, \quad \forall i \in \mathcal{T}, \forall r \in R(i) \tag{5.1f}$$

$$\sum_{r \in M(i)} m_{ir} = 1, \quad \forall i \in \mathcal{T} \tag{5.1g}$$

$$K_{ij} + K_{ji} = 1, \quad \forall (i,j) \in \mathcal{D} \tag{5.1h}$$

$$\tau \geq 0 \tag{5.1i}$$

$$u_i \geq 0, \quad \forall i \in \mathcal{T} \tag{5.1j}$$

$$K_{ij} \in \mathbb{Z}, \quad \forall (i,j) \in \mathcal{D} \tag{5.1k}$$

$$m_{ir} \in \{0,1\}, \quad \forall i \in \mathcal{T}, \quad \forall r \in R(i). \tag{5.1l}$$

Ce modèle possède le même critère que (1.8) puisque l'on cherche à maximiser le taux de production. Le modèle n'est pas détaillé ici mais le lecteur trouvera la description exhaustive des contraintes dans [Quinton 2020].

Exemple 10. *L'application du modèle (5.1) à l'exemple 9 produit l'ordonnancement optimal décrit en figure 5.2. Le temps de cycle obtenu est égal à 10.*

On peut remarquer que la solution de la figure 5.2 est meilleure que celle de la figure 5.1, bien que l'ordonnancement de la figure 5.1 sature la machine 1.

5.3.2 Méthode de décomposition

La formulation (5.1) du FCJSP peut être très difficile à résoudre pour des instances avec un nombre important de tâches ou peu de robots. Face à ce constat, nous

proposons une décomposition de Benders. La formulation de notre décomposition est classique, dans le sens où le problème maître correspond à un programme en variable entière composé des contraintes (5.1) mettant en jeu les variables d'affectation et les variables de décalage d'occurrence, c'est-à-dire les contraintes (5.1d) et (5.1g). Les autres contraintes, impliquant les variables continues, composent le sous problème. La première version de cette décomposition donnait des résultats médiocres. En effet, il n'y avait pas de relaxation du sous-problème et les valeurs des décalages d'occurrence et des variables d'affectation n'étaient pas guidées par la fonction objectif qui n'est constituée que de la valeur du temps de cycle. Dans cette première version, le problème maître se comportait comme "un poulet sans tête" (voir figure 5.3) puisque les variables n'étaient pas liées à la fonction objectif et il générait de nombreuses de coupes de faisabilité avant de fournir une coupe d'optimalité. Pour contourner cette difficulté, nous avons introduit une relaxation du sous-problème dans le problème maître. De cette façon, toutes les solutions proposées par le problème maître étaient réalisables et les coupes produites par le sous-problème étaient donc des coupes d'optimalité. Afin d'améliorer l'efficacité de cette méthode, nous avons aussi utilisé la technique des "points coeur" de [Magnanti 1981].

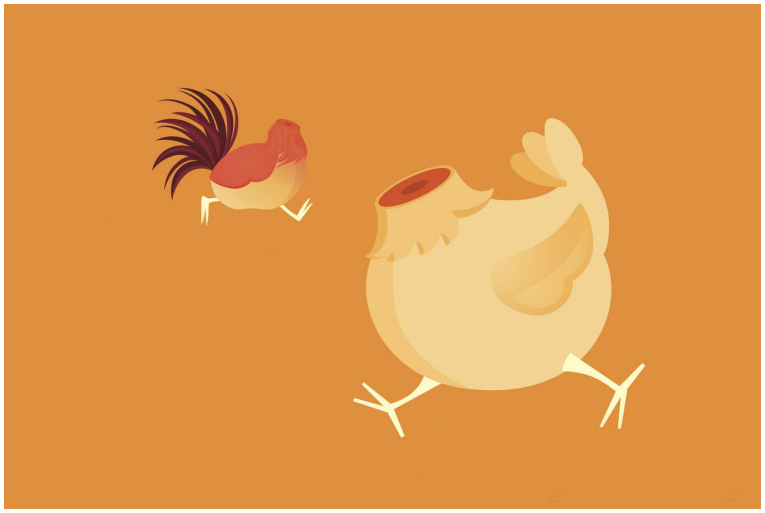


FIGURE 5.3 – Un poulet sans tête.

Exemple 11. *Le résultat de la décomposition de Benders sur l'exemple 9 correspond à l'ordonnancement de la figure 5.4. On peut remarquer que la solution obtenue est différente de la solution précédente (MIP) avec un temps de cycle identique.*

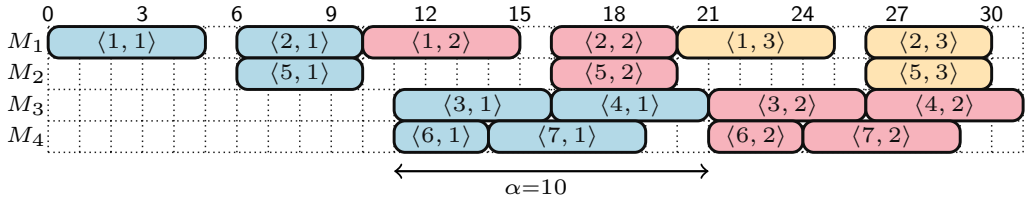


FIGURE 5.4 – Ordonnancement produit par la décomposition de Benders.

5.3.3 Méthodes approchées

Le modèle exposé dans la section précédente atteint rapidement ses limites (voir les résultats expérimentaux). Par conséquent, une heuristiques est développée. Elle consiste à réduire la flexibilité du problème en restreignant le nombre de machines candidates à l'exécution d'une tâche.

Dans cette approche, nous cherchons à réduire la flexibilité des tâches, c'est-à-dire le nombre de machines qui peuvent exécuter une tâche donnée. On définit $R^H(i) \subset R(i)$ l'ensemble restreint des ressources qui peuvent exécuter la tâche i. Cette approche est appelée "$q-$Flex", où $q = card(R^H(i))$. Le programme linéaire en nombre entier (5.2a-5.2d) permet de construire les ensembles $R^H(i)$ de telle sorte que le temps total de travail des machines soit équilibré entre toutes les machines.

$$\min P_{max}$$

s.t.

$$P_{max} \geq \sum_{i \in \mathcal{T}} m_{ir} p_{ir}, \quad \forall r \in R \tag{5.2a}$$

$$\sum_{r \in r(i)} m_{ir} = q, \quad \forall i \in \mathcal{T} \tag{5.2b}$$

$$P_{max} \in \mathbb{R} \tag{5.2c}$$

$$m_{ir} \in \{0, 1\}, \quad \forall i \in \mathcal{T}, \forall r \in R(i). \tag{5.2d}$$

Pour que cette heuristique soit efficace, il faut évidemment choisir q tel que $q < card(R(i))$. Les résultats avec $q = 2$ semblent être un bon compromis entre qualité de la solution et temps de calcul réduit mais l'heuristique est paramétrable et une valeur plus importante de q peut produire une meilleure solution au prix d'un temps calcul plus grand.

Exemple 12. *En appliquant cette heuristique avec $q = 2$ à l'exemple 9, nous obtenons le temps de cycle optimal $\alpha = 10$, cependant, l'affectation des tâches aux machines diffère de celle qui a été trouvée par le MIP et la décomposition de Benders. Le nouvel ordonnancement est présenté en figure 5.5.*

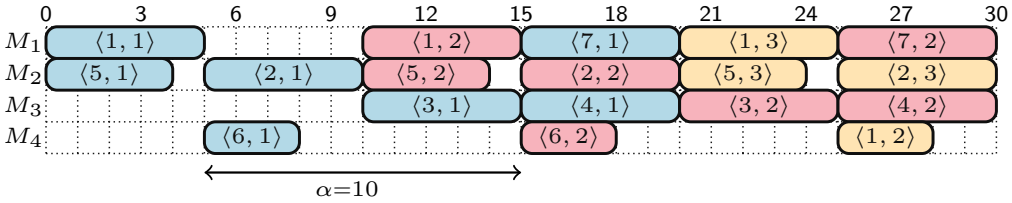


FIGURE 5.5 – Ordonnancement produit par l'heuristique basée sur la réduction de la flexibilité.

5.4 Expérimentations numériques

Des expérimentations numériques ont été conduites sur des instances générées aléatoirement. Le tableau 5.2 reporte les temps de calcul des différentes méthodes de résolution du FCJSP sur 30 instances de 10 tâches avec 3, 4 et 5 ressources. Dans la colonne concernant l'heuristique, le gap par rapport à la meilleure solution est mentionnée. Le MIP est très efficace pour les instances les plus faciles (5 machines). Pour ce type d'instance, il n'a pas d'adversaire. Concernant les instances de difficulté moyenne (4 machines), le MIP est toujours le plus efficace mais son temps de résolution a considérablement augmenté par rapport à la méthode de Benders. Dans ces conditions, la méthode approchée fournit rapidement des résultats souvent proches de l'optimum. Pour les instances les plus difficiles, la méthode de Benders est la plus rapide à trouver la meilleure solution. Cependant, l'heuristique est également très efficace, produisant très souvent la meilleure solution promptement.

La figure 5.6 présente l'écart relatif entre la meilleure solution et la solution donnée par l'heuristique. Nous pouvons en déduire que pour les instances les plus difficiles, cette méthode fournit une solution de très bonne qualité dans un temps raisonnable.

5.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à un problème d'ordonnancement cyclique avec machines flexibles. Le problème semble assez proche du CJSP mais le modèle mathématique proposé est finalement bien différent et moins compact que celui du CJSP. Une méthode de Benders et une méthode heuristique ont également été introduites.

FIGURE 5.6 – Gap moyen entre la solution optimale et la solution générée par la procédure heuristique.

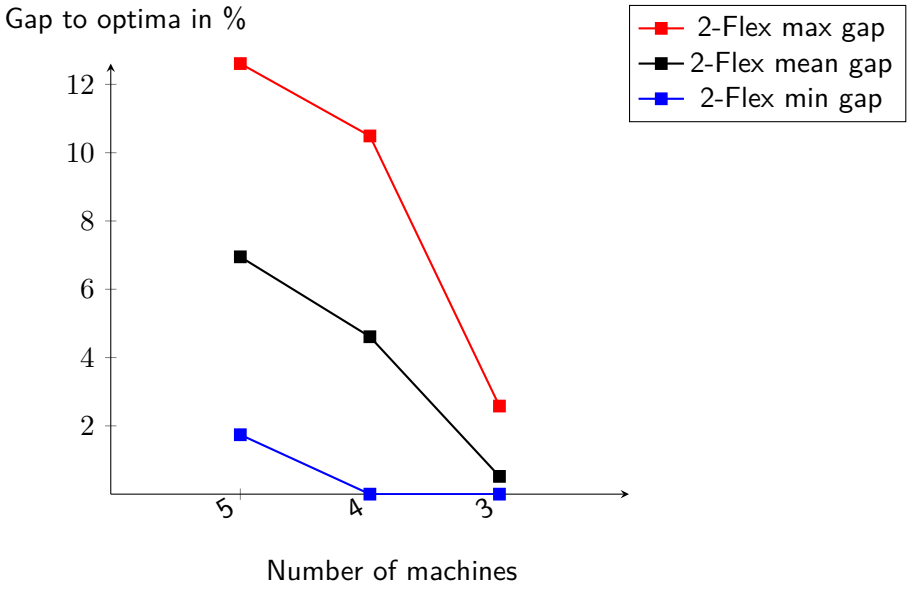


TABLE 5.2 – Comparaison du temps moyen (sec) de résolution entre le MIP et les heuristiques.

Instance name	MILP	Benders decomposition	2-Flex heuristic
inst0_10tasks_5machines	0.76	292.75	0.27(4.57%)
inst1_10tasks_5machines	1.11	459.37	0.48(9.28%)
inst2_10tasks_5machines	0.15	342.91	0.52(12.61%)
inst3_10tasks_5machines	2.98	455.63	0.53(9.01%)
inst4_10tasks_5machines	0.44	336.79	0.17(3.35%)
inst5_10tasks_5machines	0.14	353.07	0.15(1.74%)
inst6_10tasks_5machines	5.78	353.23	0.46(10.21%)
inst7_10tasks_5machines	2.38	451.89	0.10(1.86%)
inst8_10tasks_5machines	0.83	470.23	0.20(10.37%)
inst9_10tasks_5machines	0.91	332.39	0.44(6.55%)
inst0_10tasks_4machines	29.37	505.57	1.78(10.49%)
inst1_10tasks_4machines	461.93	668.82	0.63(-)
inst2_10tasks_4machines	52.68	356.89	0.91(8.25%)
inst3_10tasks_4machines	132.53	703.93	1.04(2.53%)
inst4_10tasks_4machines	93.66	458.89	1.01(1.40%)
inst5_10tasks_4machines	19.78	364.25	0.43(5.44%)
inst6_10tasks_4machines	9.15	514.52	0.66(9.57%)
inst7_10tasks_4machines	49.81	455.74	0.31(2.97%)
inst8_10tasks_4machines	124.83	659.4	0.86(2.46%)
inst9_10tasks_4machines	14.65	317.62	0.87(2.97%)
inst0_10tasks_3machines	timeout	819.95	66.29(-)
inst1_10tasks_3machines	946.08	614.28	2.93(-)
inst2_10tasks_3machines	1640.75	516.42	6.74(-)
inst3_10tasks_3machines	823.53	589.63	27.07(-)
inst4_10tasks_3machines	658.47	385.83	34.43(2.58%)
inst5_10tasks_3machines	765.6	424.89	21.01(2.58%)
inst6_10tasks_3machines	timeout	954.92	7.62(-)
inst7_10tasks_3machines	2383.02	945.99	34.74(-)
inst8_10tasks_3machines	3215.08	769.91	9.81(-)
inst9_10tasks_3machines	2337.64	473.86	13.17(-)

Deuxième partie

Optimisation pour le spatial

Chapitre 6

Allocation de fréquence pour les systèmes de satellite SDMA

Sommaire

6.1 Introduction 73
6.2 Système de télécommunication SDMA 74
6.2.1 Description des contraintes du système 76
6.2.2 Description des scénarios 76
6.3 Résolution et résultats expérimentaux. 77
6.3.1 Scénario 1 77
6.3.2 Scénario 2 77
6.4 Méthodes avec décentrage de faisceaux. 80
6.4.1 Description de la méthode 80
6.4.2 Résultats expérimentaux 82
6.5 Conclusion 84

6.1 Introduction

Dans les systèmes de télécommunication par satellite, le SDMA (Spatial Division Multiple Access) est un principe de partage de la ressource radio (RRM : Radio Resource Management) utilisant la formation de faisceaux. Plus précisément, c'est le couplage d'une allocation flexible de la ressource et de la formation de faisceaux adaptative. Cette technologie permet de choisir l'emplacement des faisceaux sur une zone de service, contrairement à une utilisation classique où les faisceaux sont régulièrement répartis. Cependant, cette nouvelle technologie ne va pas sans poser de nouveaux problèmes d'optimisation. En effet, l'affectation des fréquences reste un problème difficile à résoudre par la nature elle-même du problème (le problème de décision associé au problème de coloration d'un graphe est NP-complet). Par ailleurs, le placement lui-même des faisceaux est un problème d'optimisation non-linéaire puisque la forme des faisceaux n'est pas concave. Dans ce chapitre, je m'intéresse à l'intérêt d'utiliser cette technologie récente pour les systèmes de télécommunication par satellite. Mon intérêt pour cette thématique est apparu avec la réalisation de deux collaborations industrielles avec Thales Alenia Space en 2009 et 2010. L'objectif était alors de quantifier l'intérêt de centrer les faisceaux sur les utilisateurs par rapport à un pavage régulier de la zone de service.

Le travail de ce chapitre a fait l'objet des publications [Houssin 2011b], [Kiatmanaroj 2012a], [Kiatmanaroj 2012b] et [Kiatmanaroj 2013]. Une partie de ce chapitre est issue d'une collaboration industrielle avec Thales Alenia Space et une collaboration académique avec Frédéric Messine.

Après une introduction, nous présentons un système de télécommunication utilisant la technologie SDMA dans la section 6.2.

6.2 Système de télécommunication SDMA

L'objectif est ici de servir en fréquence un maximum d'utilisateurs distribués uniformément dans une zone de service. Les fréquences peuvent être considérées comme des couleurs car les utilisateurs demandent tous une bande passante de taille identique.

Les faisceaux ont deux caractéristiques particulières qui sont la position du centre du faisceau dans la zone de service et le diagramme de rayonnement. Une représentation analytique du diagramme de rayonnement permet de calculer le gain directif des antennes pour une direction considérée. La fonction décrivant le diagramme de rayonnement est présentée dans [Houssin 2011b], elle est fortement non linéaire et présente des lobs secondaires. Des représentations de diagramme de rayonnement sont exposées en figures 6.1 et 6.2. Le diagramme est fonction de la position du centre du faisceau et de la position satellite. Dans les figures 6.1 et 6.2, un diagramme est centré directement en dessous du satellite (au point $(0, 0)$ et un autre est centré sur $(0.03, -0.03)$. Sans faire appel à la fonction analytique du gain, deux observations peuvent être établies :

— Plus le point de la zone de service est éloigné de l'aplomb du satellite (point défini par les coordonnées $(0, 0)$), moins le gain est élevé.
— Des lobs secondaires sont présents autour du lob principal.

Ces lobs secondaires ont un effet important puisque dans le cas où des utilisateurs ont la même fréquence, les lobs secondaires des uns vont affecter le rapport signal sur interférences des autres.

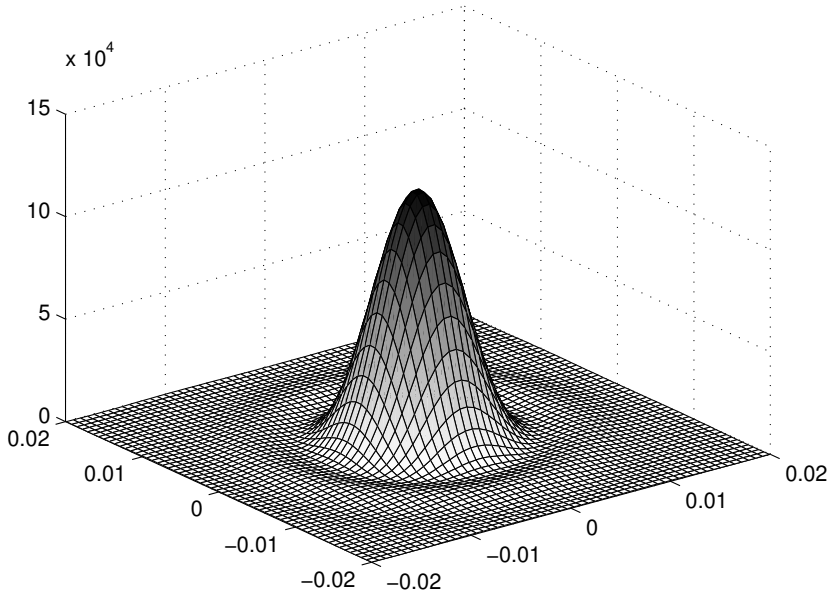


FIGURE 6.1 – Représentation 3D d'un faisceau d'antenne.

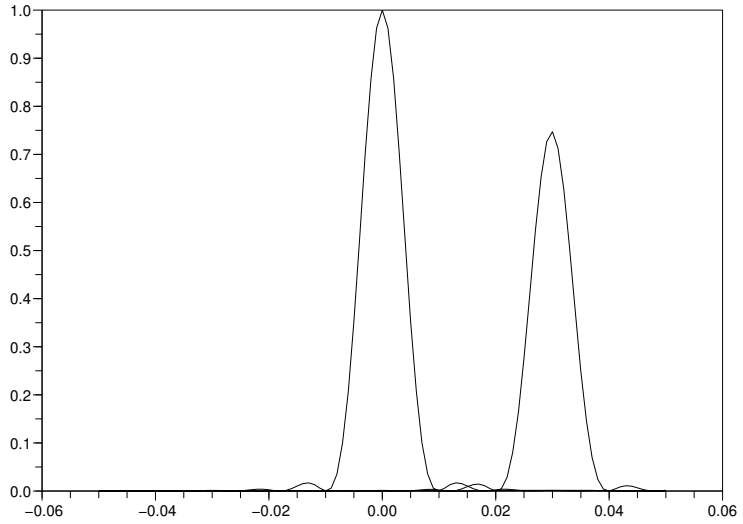


FIGURE 6.2 – Représentation 2D des faisceaux d'antenne.

6.2.1 Description des contraintes du système

Sans entrer dans le détail des contraintes, nous essayons ici de faire une description grossière mais fidèle de celles-ci. D'une part, le rapport signal sur bruit plus interférence d'un utilisateur, noté $\frac{C}{N+I}$, doit être supérieur à un certain niveau (connu) pour qu'il soit servi. Les rapports $\frac{C}{N+I}$ dépendent du rapport signal sur bruit et du rapport signal sur interférence, respectivement notés $\frac{C}{N}$ et $\frac{C}{I}$. Le rapport signal sur bruit d'un utilisateur peut être calculé en amont de la phase d'optimisation (toutes les positions sont connus). Néanmoins, le rapport signal sur interférences d'un utilisateur est dépendant de l'affectation de la fréquence aux autres utilisateurs. Si des utilisateurs proches d'un utilisateur donné, ont la même fréquence que lui, ce dernier va souffrir des interférences causées par ses voisins. Ces interférences sont cumulatives. Ce principe est illustré par la figure 6.3. Considérons que les utilisateurs u_1, u_2 et u_3 possèdent la même fréquence. L'utilisateur u_1 a un faisceau qui pointe directement sur lui, son rapport $\frac{C}{N}$ est donc très bon, cependant u_2 et u_3 (qui ont aussi leur propres faisceaux, non représentés sur la figure) vont interférer le signal émis par u_1 et détériorer le ratio $\frac{C}{I}$ de u_1. Cela est d'autant plus vrai que u_2 et u_3 se trouve sur des lobs secondaires du faisceau qui pointe sur u_1. L'utilisateur u_1 subit donc l'effet cumulé des interférences causées par u_2 et u_3.

D'autre part, on ne peut affecter à un utilisateur qu'au plus une seule couleur.

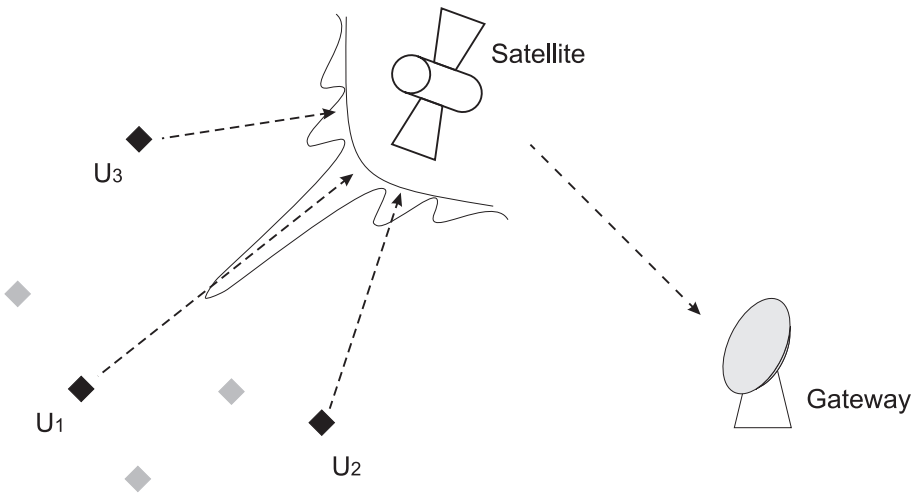


FIGURE 6.3 – Lien utilisateurs vers passerelle.

6.2.2 Description des scénarios

Dans cette partie, nous avons considéré deux scénarios : un scénario (noté scénario 1) où chaque utilisateur possède un faisceau centré sur lui, le gain est alors maximal, et un scénario où les faisceaux sont fixés par un motif régulier. A chaque faisceau, 8 couleurs peuvent être affectées. Par conséquent dans le scénario 2, un unique faisceau peut servir jusqu'à 8 utilisateurs. Dans les deux cas, le problème est statique : l'ensemble des utilisateurs est connu à l'avance et il n'y a ni arrivée, ni départ.

6.3 Résolution et résultats expérimentaux.

Beaucoup d'approches qui traitent des problèmes d'affectation de fréquence considèrent des contraintes d'interférence binaires impliquant seulement deux utilisateurs ce qui rapproche ce problème de celui de coloration d'un graphe (NP-difficile). Dans ce contexte, le panel des méthodes de résolution est large : recherche locale, méta-heuristique, MIP, programmation par contraintes sont des stratégies régulièrement rencontrées (voir [Murphey 1999, Aardal 2003, Montemanni 2003, Resende 2006]). Dans cette étude, les interférences ne sont pas binaires mais cumulatives. En fait, un utilisateur peut avoir une fréquence si son niveau cumulé d'interférence est inférieur à un seuil qui dépend de la position de l'utilisateur. Dans la littérature, peu de travaux prennent en compte cette caractéristique : [Dunkin 1998, Mannino 2003, Palpant 2008].

Pour chaque scénario décrit précédemment, 2 MIP, très similaires et détaillés dans [Houssin 2011b], ont été implémentés ainsi qu'une méthode heuristique, non présentée ici mais disponible dans [Houssin 2011b], basée sur l'algorithme DSATUR (voir [Brélaz 1979]).

6.3.1 Scénario 1

Dans ce scénario, chaque utilisateur possède un faisceau centré sur lui. Le MIP présenté ci-dessous correspond à la résolution du scénario 1.

$$\max \sum_{i=1}^{n} \sum_{c=1}^{C} x_{ic} \tag{6.1}$$

$$\sum_{c=1}^{C} x_{ic} \leq 1 \qquad i = 1, \ldots, n \tag{6.2}$$

$$\sum_{j=1}^{n} \delta_{i,j} x_{jc} \leq \alpha_i + M_i(1 - x_{ic}) \qquad i = 1, \ldots, n \quad c = 1, \ldots, C \tag{6.3}$$

$$x_{ic} \in \{0, 1\} \qquad i = 1, \ldots, n \quad c = 1, \ldots, C. \tag{6.4}$$

La variable binaire x_{ic} vaut 1 si la fréquence c est affectée à l'utilisateur i , 0 sinon. On cherche à maximiser le nombre d'utilisateurs servis et un utilisateur peut avoir au plus une seule fréquence. L'inégalité (6.3) est la contrainte d'interférence cumulative. Le terme α_i peut être vu comme le niveau acceptable d'interférence pour l'utilisateur i et $\delta_{i,j}$ comme l'effet cumulatif de l'interférence de l'utilisateur j sur l'utilisateur i s'ils ont la même fréquence, sinon la contrainte est relâchée. Les paramètres α_i et $\delta_{i,j}$ peuvent facilement être calculés en amont.

6.3.2 Scénario 2

Le modèle du scénario 2 est très semblable à celui du scénario 1. L'ensemble $S = \{1, \ldots, m\}$ désigne l'ensemble des faisceaux et U_s est l'ensemble des utilisateurs couverts par le faisceau $s \in S$. Un utilisateur i appartient à l'ensemble U_s correspondant

au faisceau le plus proche de lui. La contrainte (6.7) spécifie qu'une fréquence ne peut être affectée qu'une seule fois à l'intérieur d'un faisceau. La contrainte (6.8) joue le même rôle que la contrainte (6.3) du scénario 1 à la différence que les valeurs de β_i and γ_{ij} sont différentes de α_i et δ_{ij} puisque les faisceaux ne sont pas centrés sur les utilisateurs.

$$\max \sum_{i=1}^{n} \sum_{c=1}^{C} x_{ic} \tag{6.5}$$

$$\sum_{c=1}^{C} x_{ic} \leq 1 \qquad i = 1, \ldots, n \tag{6.6}$$

$$\sum_{i \in U_s} x_{ic} \leq 1 \qquad s = 1, \ldots, m \quad c = 1, \ldots, C \tag{6.7}$$

$$\sum_{j=1}^{n} \gamma_{i,j} x_{jc} \leq \beta_i + N_i(1 - x_{ic}) \qquad i = 1, \ldots, n \quad c = 1, \ldots, C \tag{6.8}$$

$$x_{ic} \in \{0, 1\} \qquad i = 1, \ldots, n \quad c = 1, \ldots, C. \tag{6.9}$$

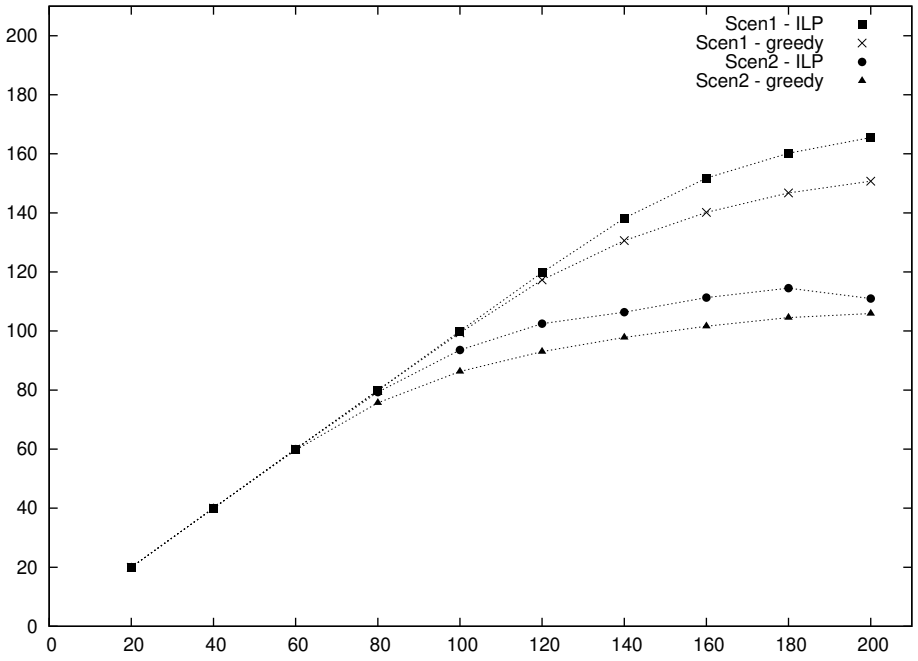


FIGURE 6.4 – Nombre moyen d'utilisateurs acceptés pour chaque nombre d'utilisateurs.

n	20	40	60	80	100	120	140	160	180	200
Scen 1 - ILP	100	100	100	100	99	94	46	3	0	0
Scen 2 - ILP	100	100	100	82	6	0	0	0	0	0

TABLE 6.1 – Nombre d'optima trouvés par le MIP.

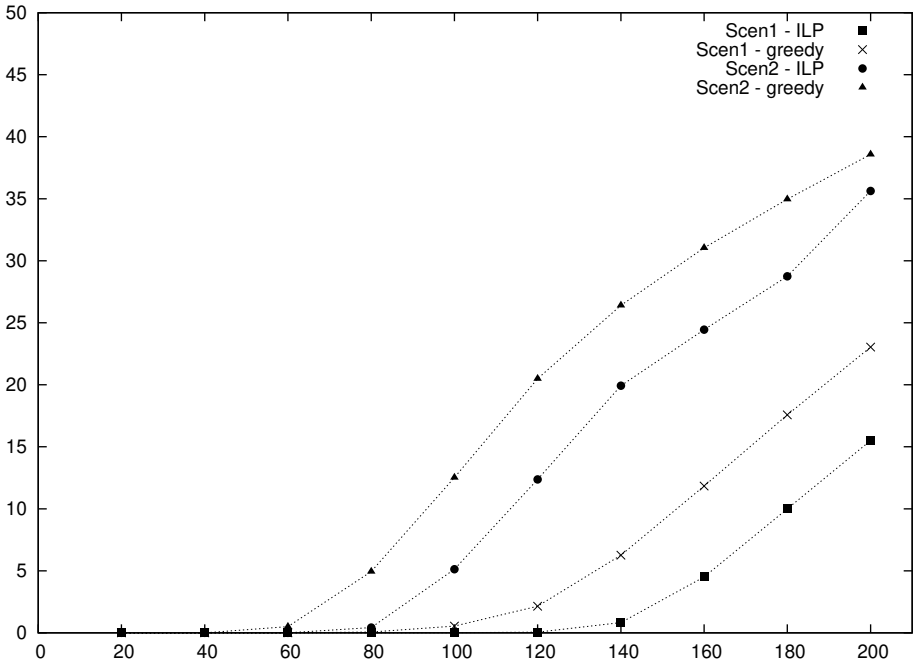


FIGURE 6.5 – Gap moyen pour chaque nombre d'utilisateurs.

Un aperçu des résultats est disponible sur la figure 6.4 où, pour chaque scénario, nous reportons le nombre moyen d'utilisateurs acceptés en fonction du nombre total d'utilisateurs présents sur la zone de service. En outre, la figure 6.5 présente le gap relatif moyen par rapport à la borne supérieur du MIP.

La figure 6.4 montre clairement que les meilleurs résultats en termes de qualité de service sont donnés par le scénario 1, indépendamment de la méthode. Le MIP obtient des résultats significativement meilleurs que l'heuristique, même pour des temps de calcul réduit (60 sec. dons notre cas). Les résultats du scénario 1 sont toujours supérieurs à ceux du scénario 2, même pour le cas où l'heuristique est utilisé pour le scénario 1 et le MIP pour le scénario 2. Cela montre bien l'intérêt de centrer les faisceaux sur les utilisateurs.

Comme attendu, la figure 6.5 indique bien que le gap augmente quand le nombre d'utilisateurs croît. Le gap est également plus important pour le scénario 2 que pour le scénario 1. Cela montre que, pour notre formulation, le problème d'optimisation du scénario 1 est plus simple à résoudre que celui posé par le scénario 2. Cela est confirmé par le nombre d'optima donné par le tableau 6.1. En effet, le tableau 6.1 montre que le

nombre d'optima trouvés chute de 100% à 82% puis 6 % quand le nombre d'utilisateurs passe de 60 à 100 pour le scénario 2 tandis que ce nombre reste au delà de 90% jusqu'à 120 utilisateurs pour le scénario 1. Interdire l'affectation de couleurs identiques à l'intérieur d'un même faisceau rend la recherche de solution optimale plus difficile.

L'analyse croisée de la figure 6.4 et du tableau 6.1 montrent que pour les instances où 100% des optima sont trouvés, c'est-à-dire pour les instances jusqu'à 60 utilisateurs, il n'y a pas de différence entre les scénarios puisque tous les utilisateurs sont acceptés. Au delà de 80 utilisateurs, les résultats du scénario 2 chutent drastiquement et même l'heuristique du scénario 1 est meilleure dans ces conditions.

Cet exemple numérique montre bien le bénéfice du centrage de faisceau sur les utilisateurs par rapport à un schema régulier classique. On peut aussi remarquer qu'à partir de 80 utilisateurs, ce qui correspond à un facteur de réutilisation de 10, les différences entre les 2 scénarios augmentent fortement.

6.4 Méthodes avec décentrage de faisceaux.

Dans la section 6.3, nous avons montré l'apport du centrage de faisceau par rapport à un motif régulier. Toujours dans cette optique, nous proposons ici une méthode qui couple le modèle du scénario 1 (défini dans §6.3.1) et une méthode d'optimisation continue pour déplacer certains faisceaux. L'intérêt de cette stratégie est de d'affecter une fréquence à des utilisateurs qui n'ont pas été servis lors de la phase MIP.

A titre d'exemple, la figure 6.6 présente un exemple d'affectation de fréquence avec 5 utilisateurs sur la zone de service. Parmi les utilisateurs, 4 sont servis par les fréquences 1 et 2 (indiquée entre parenthèse sur la figure). L'utilisateur 4 n'est pas servi (indiqué par la fréquence 0 sur la figure 6.6).

Cependant, un léger déplacement des faisceaux des utilisateurs 1, 2 et 4 permet de servir ces 3 utilisateurs avec la fréquence 1. Cette solution est illustrée par la figure 6.7.

6.4.1 Description de la méthode

Le principe général est un algorithme itératif (décrit exhaustivement dans [Kiatmanaroj 2013]) dont la première étape est le MIP du scénario 1 (il est également possible d'utiliser la méthode heuristique) puis la seconde étape correspond à une optimisation non linéaire du déplacement de faisceau. Ensuite, on réitère la procédure jusqu'à stabilisation du déplacement. Seule la seconde étape est décrite dans la suite puisque la première étape correspond au §6.3.1.

La procédure de décentrage de faisceaux prend en paramètre les solutions données par le MIP, c'est-à-dire une affectation des fréquences aux utilisateurs. Soit i un utilisateur non servi, la procédure sélectionne une couleur c et identifie un ensemble d'utilisateurs interférants tels que $x_{jc} = 1$. Les k utilisateurs les plus interférants sont sélectionnés et leur faisceau est déplacé continuellement de telle sorte que

— ces k utilisateurs soient toujours servis par la fréquence c,
— l'utilisateur i soit servi par la fréquence c,
— la somme des déplacements (distance euclidienne) soit minimisée.

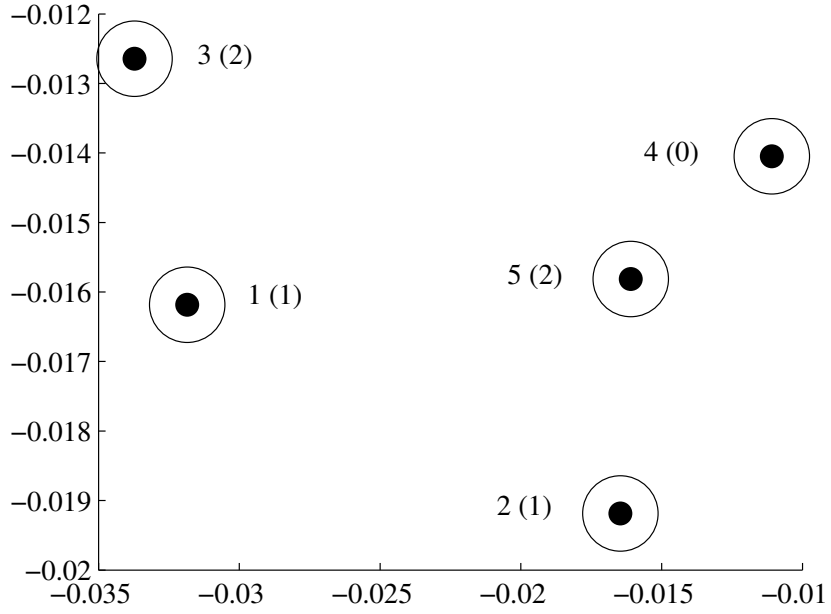


FIGURE 6.6 – Un exemple d'affectation de fréquence.

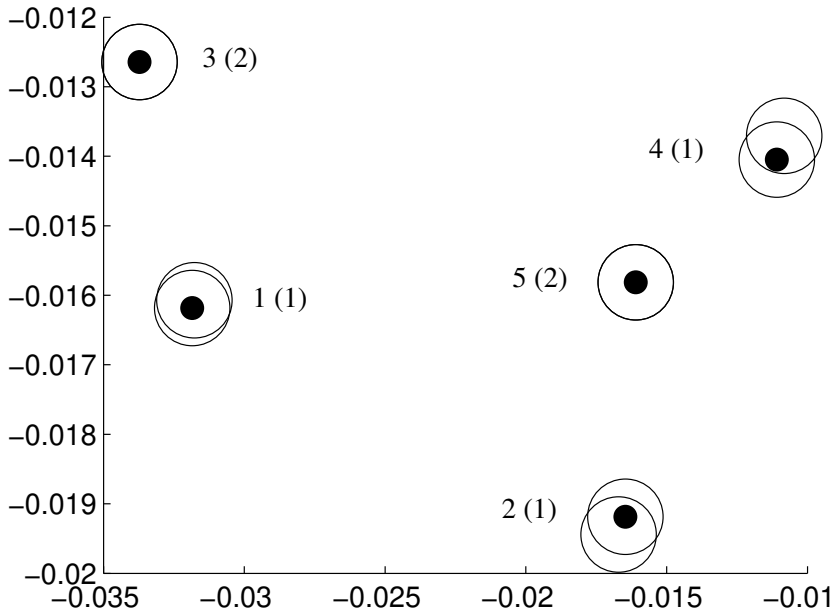


FIGURE 6.7 – Amélioration de l'affectation des fréquence après un décentrage des faisceaux.

Si on ne trouve pas de solution au problème non linéaire précédent, l'algorithme choisit une autre couleur et recommence. Si aucune couleur ne convient, l'utilisateur i est rejeté et l'algorithme passe à un autre utilisateur. Cette optimisation non-linéaire converge vers un optimum local et d'autres méthodes, qui n'ont pas été testée, telle que des métaheuristiques, permettraient probablement d'obtenir une solution plus qualitative.

6.4.2 Résultats expérimentaux

La figure 6.8 montre le nombre moyen d'utilisateurs réaffectés grâce au décentrage de faisceau pour 20 instances de 200 utilisateurs. Les valeurs de k varient de 3 à 10 ($MAXINEG$ et $UTVAR$ sont des paramètres de l'algorithme non détaillés ici). Comme attendu, plus k est grand, plus les calculs sont longs mais plus on a la possiblité de réaffecter des utilisateurs qui étaient rejetés par le MIP.

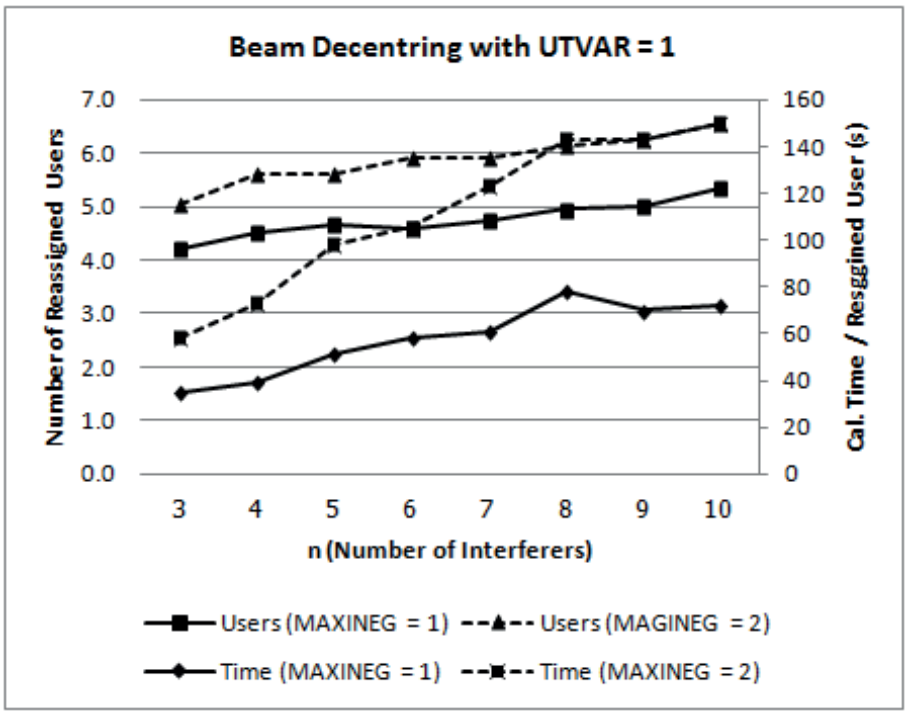


FIGURE 6.8 – Nombre moyen d'utilisateurs réaffectés et temps de calcul par réaffectation pour différentes configurations de l'algorithme.

Pour chaque algorithme et pour plusieurs nombres d'utilisateurs, les figures 6.9 et 6.10 présentent le nombre d'utilisateur acceptés. Toutes les méthodes sont équivalentes jusqu'à 120 utilisateurs. A partir de 140 utilisateurs, les différences entre les méthodes croissent et à partir de 180 utlisateurs, les performances du MIP se dégradent s'il n'a pas un temps de calcul suffisant. Dans tous les cas, à partir de 140 utilisateurs, le décentrage de faisceau a un effet bénéfique puisque le nombre d'utilisateurs servis est

sensiblement amélioré.

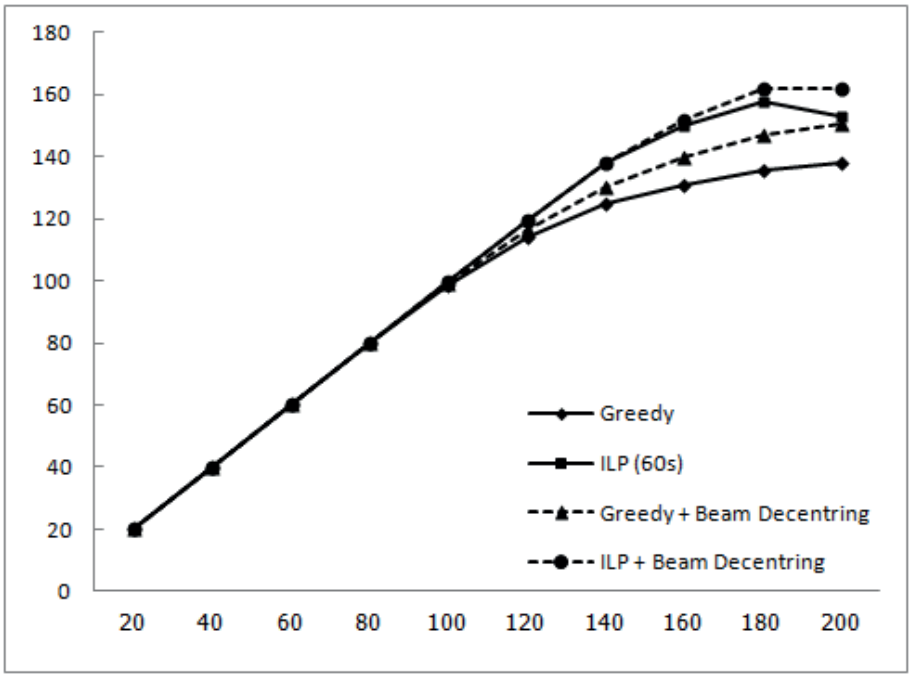


FIGURE 6.9 – Nombre moyen d'utilisateurs acceptés avant et après la procédure de décentrage en fonction du nombre d'utilisateurs pour différents algorithmes de la phase 1.

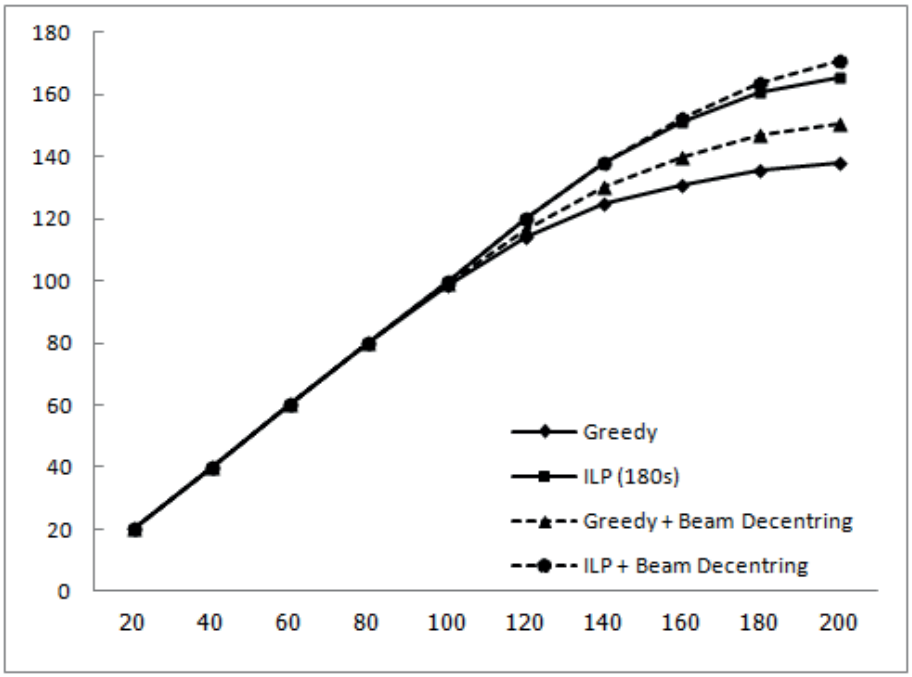


FIGURE 6.10 – Nombre moyen d'utilisateurs acceptés avant et après la procédure de décentrage pour différents algorithmes de la phase 1.

6.5 Conclusion

Dans ce chapitre, nous avons quantifié l'intérêt d'une répartition non régulière des faisceaux avec des motifs non réguliers pour la répartition de fréquence. En effet, la technologie SDMA permet de centrer un faisceau directement sur l'utilisateur, ce qui maximise le gain de son signal. Nous avons également montré que dans certains cas, un léger décentrage des faisceaux pouvait être avantageux.

Chapitre 7

Modèle d'ordonnancement pour le problème d'affectation de fréquence

Sommaire

7.1 Introduction 85
7.2 Modèles d'ordonnancement pour les problèmes d'affectation de plage de fréquence 86
7.3 Expérimentations numériques 87
7.3.1 Intérêt de la formulation en clique maximale 87
7.4 Conclusion 88

7.1 Introduction

Dans ce chapitre, nous considérons un problème d'affectation de fréquence avec deux caractéristiques majeures :

— Un utilisateur ne demande pas une seule fréquence mais un ensemble de fréquences contiguës (une bande passante).

— Des interférences binaires sont considérées. Autrement dit, deux utilisateurs interférants ne doivent pas avoir des plages de fréquence qui se chevauchent.

Le problème d'affectation de plage de fréquence est équivalent au problème de coloration d'intervalle de graphe considéré dans [de Werra 1994, Bouchard 2010, Graham 2008].

Afin de résoudre ce problème pratique, le lien entre le problème d'affectation de plage de fréquence et l'ordonnancement disjonctif est exploité. Selon [Pinedo 2008], l'ordonnancement est une prise de décision qui traite de l'allocation des ressources aux tâches sur des périodes de temps données tout en considérant que les ressources et les tâches peuvent être sujettes à des contraintes et des objectifs de nature différente. A travers cette définition, on peut voir l'ordonnancement comme un sujet vaste qui a de nombreuses applications industrielles. Dans [de Werra 1994], un lien est établi entre ordonnancement, coloration de graphe et affectation de fréquence. Plus précisément, si on considère un utilisateur comme une tâche et sa demande en fréquence comme la durée d'exécution de la tâche, maximiser le nombre d'utilisateurs servis en plage de

fréquence peut être vu comme la maximisation du nombre de tâches planifiées sous contrainte d'une date d'échéance qui correspond au nombre de fréquences disponibles. De la même façon, minimiser la plage totale de fréquence utilisée correspond à la minimisation du makespan du problème d'ordonnancement équivalent.

Ce travail a produit la publication [Kiatmanaroj 2016].

7.2 Modèles d'ordonnancement pour les problèmes d'affectation de plage de fréquence

La section précédente souligne les similarités entre les problèmes d'affectation de fréquence avec contraintes binaires et les problèmes d'ordonnancement. Dans notre cas, les interférences binaires peuvent être considérées comme des contraintes disjonctives. Pour modéliser ces contraintes disjonctives, plusieurs techniques peuvent être appliquées.

Le problème d'affectation de plage de fréquence, quelque soit le critère considéré, peut être représenté par un graphe d'interférence $G = (V, E)$ dans lequel les noeuds représentent des utilisateurs et les arcs entre deux noeuds représentent des contraintes d'interférence binaire. Chaque arc de ce graphe peut être traité comme une contrainte disjonctive ou par une contrainte de non-chevauchement en ordonnancement. Deux possibilités s'offrent à nous pour modéliser ces contraintes :

— Modéliser directement chaque arc par une contrainte disjonctive. Si le graphe possède n noeuds, il y a au plus $n(n-1)/2$ contraintes.
— Grouper les contraintes en clique maximale. Cela nécessite de trouver toutes les cliques maximales.

Dans notre cas, une clique maximale consiste en un groupe d'utilisateurs qui s'interfèrent entre eux. Pour un problème d'ordonnancement, cela revient à un ensemble de taches qui partage une unique ressource. Considérer une clique maximale est équivalent à considérer un ensemble de contraintes disjonctives simultanément, ce qui peut conduire à des techniques de résolution plus efficace. Le problème est qu'il peut y avoir un nombre exponentiel de cliques maximales dans un graphe d'interférence. Un exemple de graphe d'interférences est présenté sur la figure 7.1. Les cliques maximales de ce graphe sont $\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 5\}$ and $\{4, 5, 6\}$, ce qui donne 5 contraintes. Cependant, si on considère la modélisation directe des contraintes disjonctives, nous obtenons 7 contraintes (le graphe possède 7 arcs).

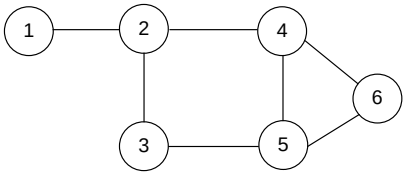


FIGURE 7.1 – Un graphe d'interférence.

Deux objectifs sont considérés pour ce problème d'affectation de plage de fréquence :

— maximisation du nombre d'utilisateurs servis,
— minimisation de la bande passante totale (minimisation de la dernière fréquence utilisée)

7.3 Expérimentations numériques

7.3.1 Intérêt de la formulation en clique maximale

Afin de valider le bénéfice de considérer les cliques maximales, nous considérons des instances difficiles pour lesquelles nous intégrons progressivement les cliques maximales. Les cliques maximales sont énumérées à l'aide de l'algorithme de Bron-Kerbosch ([Bron 1973]) et 100 instances à 500 puis 1000 utilisateurs sont étudiées pendant 60 secondes. Les instances sont résolus avec IBM CP Optimizer. Pour les instances à 500 utilisateurs, le temps moyen de recherche des cliques maximales est de l'ordre de la seconde, tandis que pour les instances à 1000 utilisateurs, le temps moyen est de 22.95 secondes.

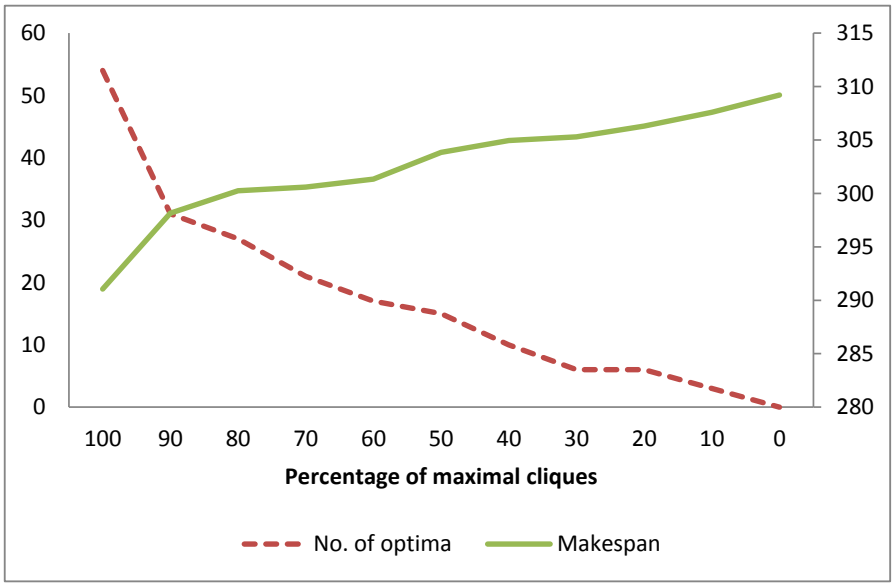


FIGURE 7.2 – Nombre d'optima moyen en fonction du pourcentage de cliques maximales utilisé avec 500 utilisateurs.

On constate sur les figures 7.2 et 7.3 que considérer toutes les cliques maximales améliorent toujours les performances. Le makespan obtenu (ou la dernière fréquence utilisée) est considérablement plus faible avec 100% des cliques maximales que sans clique maximale. La conclusion de cette sous-section est que le solveur CP n'est pas pénalisé par l'ajout de toutes les cliques maximales.

Les figures 7.4 et 7.5 corroborent le bénéfice de l'utilisation des cliques maximales sur les contraintes disjonctives en ce qui concerne la minimisation du makespan. Cependant la différence est faible concernant le makespan.

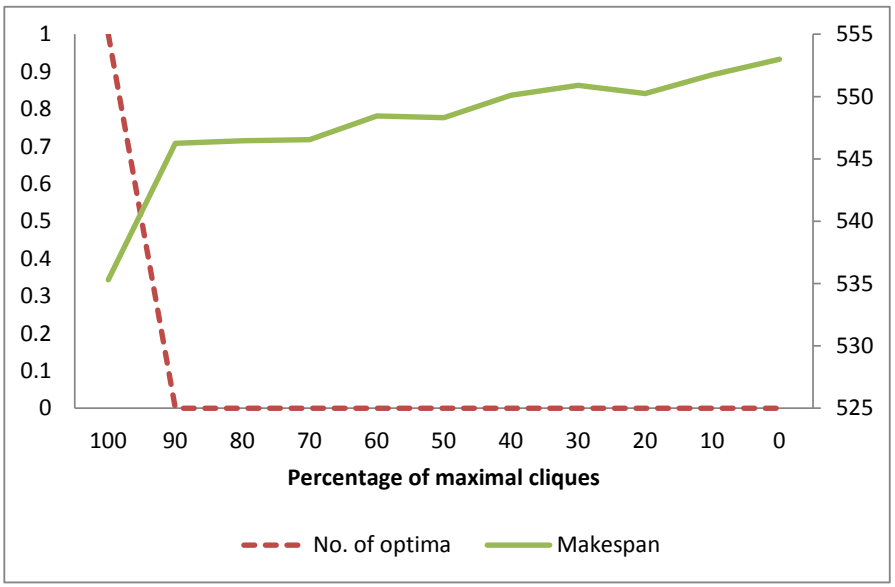


FIGURE 7.3 – Nombre d'optima moyen en fonction du pourcentage de cliques maximales utilisé avec 1000 utilisateurs.

Concernant le problème d'affectation des plages de fréquence aux utilisateurs, 2 types d'instances sont considérées suivant la taille de la bande passante : 60MHz ou 100MHz. Dans ce cas, le modèle avec les cliques maximales donnent des résultats très légèrement inférieurs en termes d'objectifs comme le montre la figure 7.6. Cependant, la figure 7.7 indique qu'en termes d'optima trouvés, le modèle avec les cliques maximales est toujours meilleur.

7.4 Conclusion

Nous avons traité dans ce chapitre le problème d'allocation de bande de fréquence avec des contraintes binaires. Nous mettons en lumière l'intérêt d'utiliser des cliques maximales dans la formulation malgré le coût important de leur recherche.

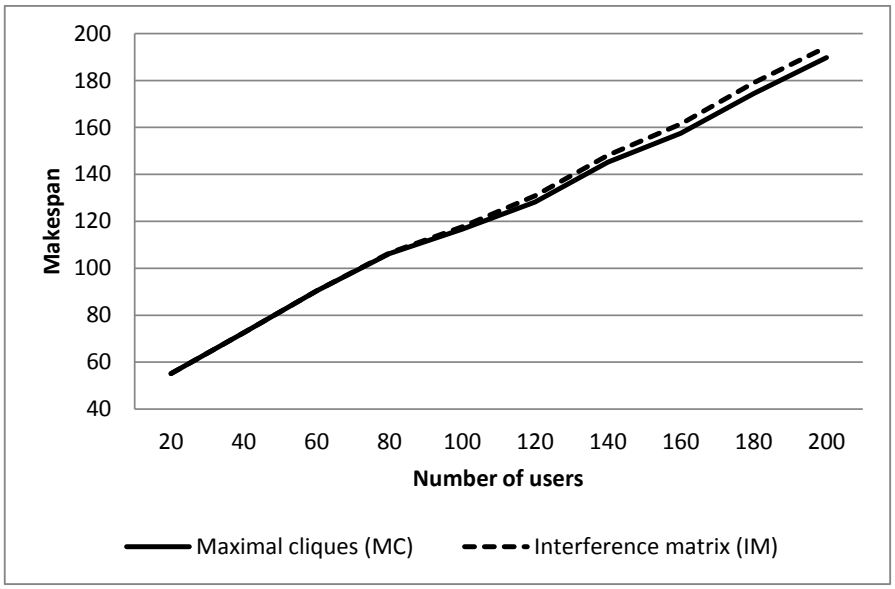


FIGURE 7.4 – Makespan en fonction du nombre d'utilisateurs.

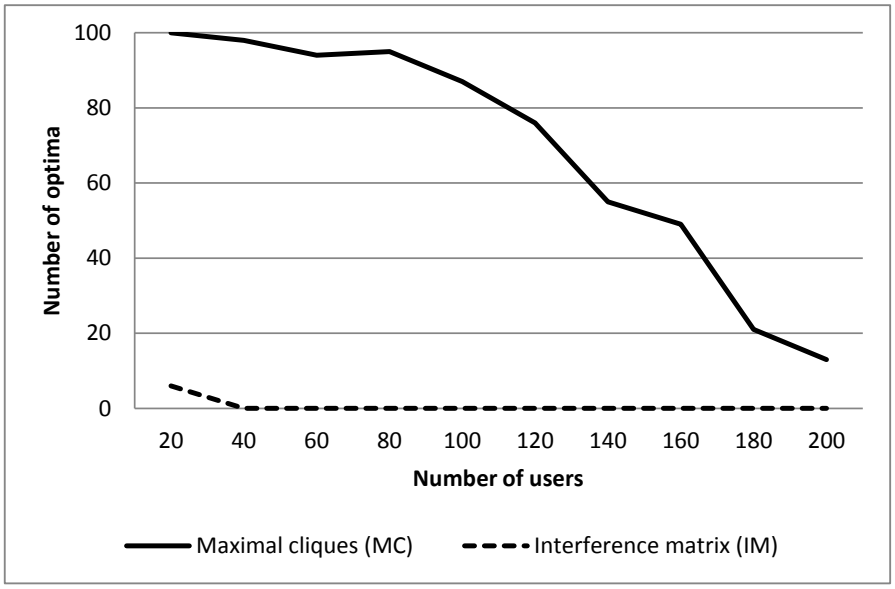


FIGURE 7.5 – Nombre d'optima en fonction du nombre d'utilisateurs.

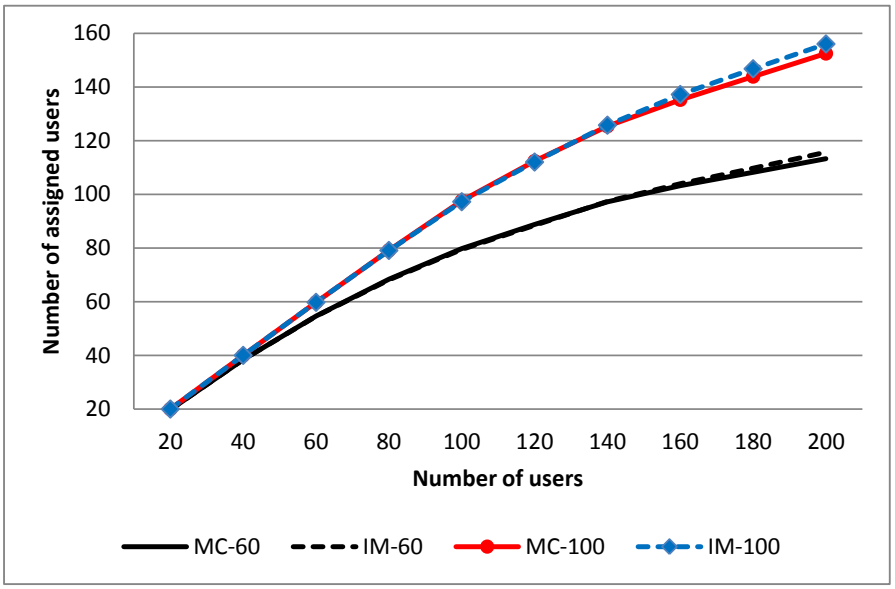


FIGURE 7.6 – Makespan en fonction du nombre d'utilisateurs.

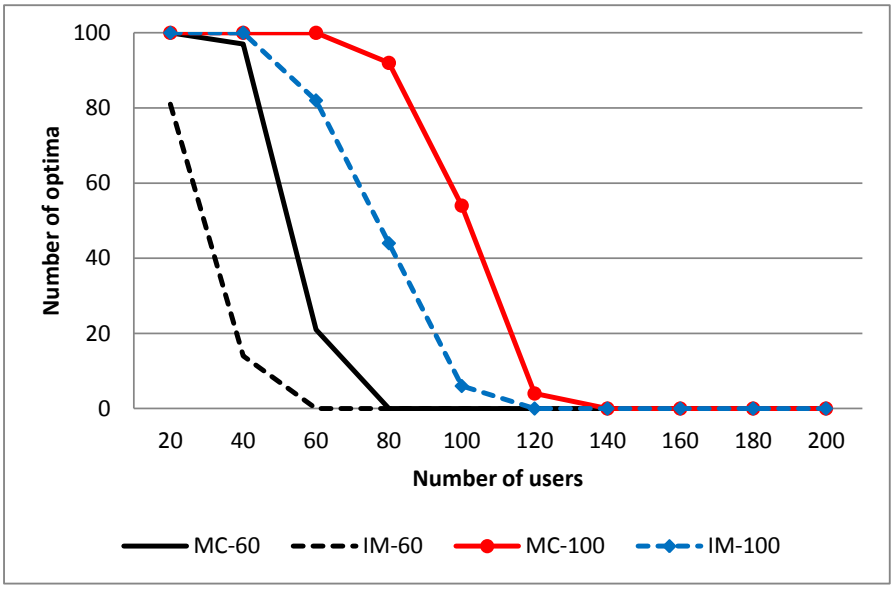


FIGURE 7.7 – Nombre d'optima en fonction du nombre d'utilisateurs.

Chapitre 8

Optimisation de la charge utile dans les satellites de télécommunication

Sommaire

8.1 Introduction 91
8.2 Linéarisation de la norme euclidienne 92
8.2.1 Encadrement de la norme euclidienne 92
8.2.2 Qualité de l'approximation 93
8.2.3 Utilisation pour le problème du $k-$centre continu 94
8.3 Application au problème de placement de faisceaux. 94
8.4 Amélioration de la formulation 98
8.5 Conclusion 99

8.1 Introduction

Ce chapitre propose d'optimiser la définition du plan de faisceaux d'un système de satellites de télécommunication multifaisceaux. Plusieurs similarités avec le chapitre précédent peuvent être mentionnées puisqu'on travaille toujours sur une zone de service limitée et que l'on doit servir des utilisateurs (ici des stations). Cependant, le problème présenté dans ce chapitre est plus proche de la réalité.

Par rapport au chapitre précédent, les interférences sont évitées en fixant une distance minimale (contrainte de séparation) entre deux centres de faisceau qui utilisent le même réflecteur.

Plus précisément, il s'agit de déterminer le centre des faisceaux des satellites, leur diamètre et leur réflecteur. Le problème est non linéaire puisqu'il met en jeu des contraintes de proximité (convexe) entre le centre de faisceau et une station et des contraintes de séparation entre 2 faisceaux qui utilisent le même réflecteur. L'objectif est de servir la demande, qui peut être vue comme une somme pondérée des stations servis. Une grande partie du problème a été simplifiée par l'introduction d'une linéarisation de la norme euclidienne. Cette linéarisation est introduite en début de ce chapitre.

Ce problème est issu de la thèse CIFRE de Jean-Thomas Camino qui a marqué une collaboration entre le LAAS-CNRS et Airbus Defence and Space. Les publica-

tions suivantes résultent de ce travail : [Camino 2015, Camino 2021, Camino 2019, Camino 2014, Camino 2016b].

8.2 Linéarisation de la norme euclidienne

8.2.1 Encadrement de la norme euclidienne

Dans [Camino 2019], nous avons exposé une linéarisation de la norme euclidienne (ou norme L_2) en dimension 2. Cette linéarisation fournit un encadrement de la norme euclidienne dont la précision est paramétrable. Sans entrer dans les détails, nous la présentons dans la suite de cette section.

Plusieurs propositions ont été faites dans la littérature pour traiter ce problème, cependant la plupart d'entre elles ne considèrent qu'une sur-approximation de la norme sans fournir d'encadrement ou alors une approximation assez précise mais qui n'est ni une sur-approximation, ni une sous-approximation, ce qui interdit son utilisation dans des contraintes d'un problème d'optimisation. Néanmoins, nous pouvons citer [Liberti 2009, D'Ambrosio 2017, Celebi 2011, Mukherjee 2013]. A noter qu'une extension de cette linéarisation a été proposée pour les ellipsoïdes dans [Duguet 2022].

Le principe de notre linéarisation repose sur une discrétisation paramétrée des directions du plan, qui sont autrement caractérisée par le domaine continue $[0, 2\pi[$. En pratique, on considère un nombre de directions noté $n_{\text{directions}}$ tel que $n_{\text{directions}} \in \mathbb{N}$ et $n_{\text{directions}} \geq 3$. La $i-$ème direction est alors notée U_i avec $i \in \{1, \ldots, n_{\text{directions}}\}$. Plus précisément, U_i est défini par

$$U_i = \begin{pmatrix} U_{i,x} \\ U_{i,y} \end{pmatrix} = \begin{pmatrix} \cos\left(\dfrac{2(i-1)\pi}{n_{\text{directions}}}\right) \\ \sin\left(\dfrac{2(i-1)\pi}{n_{\text{directions}}}\right) \end{pmatrix} \in \mathbb{R}^2. \tag{8.1}$$

Ces racines fournissent une discrétisation régulière des directions du plan euclidien, chaque direction résultante représentant un sous-intervalle exclusif de $[0, 2\pi[$ de taille $\frac{2\pi}{n_{\text{directions}}}$. La figure 8.1 fournit un exemple de ce type de discrétisation. A noter que par définition, on a :

$$\forall i \in \mathcal{U}, \quad \|U_i\| = 1. \tag{8.2}$$

Les 2 propositions suivantes (non démontrées ici mais disponibles dans [Camino 2019]) permettent de caractériser l'encadrement. La première proposition permet de déterminer si deux points $u, v \in \mathbb{R}^2$ sont plus proches d'une distance donnée d. Il faut alors vérifier que toutes les projections du vecteur $u - v$ sur les vecteurs U_i sont toutes inférieures à $d \in \mathbb{R}^+$. La linéarisation de la proposition 3 donne le polygone $\mathcal{P}'$.

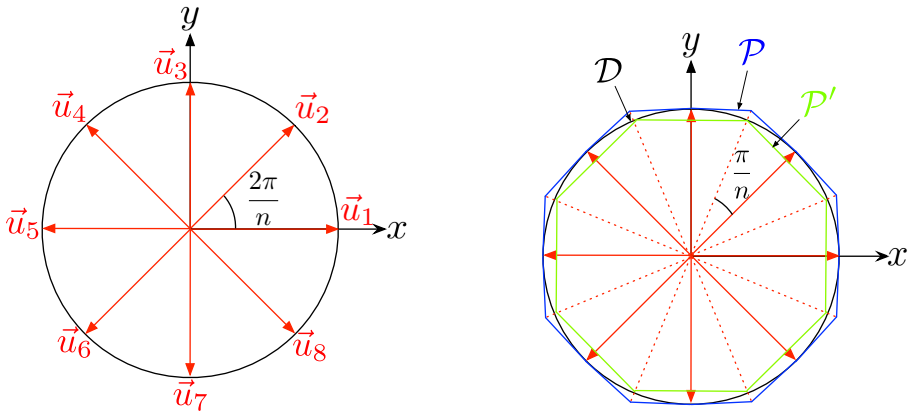


FIGURE 8.1 – (a) Discrétisation des directions du plan euclidien pour $n_{\text{directions}} = 8$. (b) Approximation du disque $\mathcal{D}$ du plan euclidien par les polygones réguliers $\mathcal{P}$ et $\mathcal{P}'$ à $n_{\text{directions}}$ côtés.

Proposition 3. *Soient $u, v \in \mathbb{R}^2$ et $d \in \mathbb{R}^+$,*

$$[\ \forall i \in \mathcal{U},\ \ \langle u - v \mid U_i \rangle \leq d.\cos(\theta_{max})\ ] \Longrightarrow \|u - v\| \leq d \tag{8.3}$$

où $\theta_{max} = \dfrac{\pi}{n_{directions}}$.

En complément de la proposition précédente, la Proposition 4 définit une procédure linéaire pour vérifier cette fois que deux points $u, v \in \mathbb{R}^2$ sont suffisamment séparés, étant donné une distance requise de séparation $d \in \mathbb{R}^+$. Cette contrainte est non-convexe. D'un point de vue pratique, cela signifie qu'il suffit de trouver, parmi les $n_{\text{directions}}$ directions discrétisées, une direction pour laquelle le produit scalaire $\langle u - v \mid U_i \rangle$ est suffisamment grand.

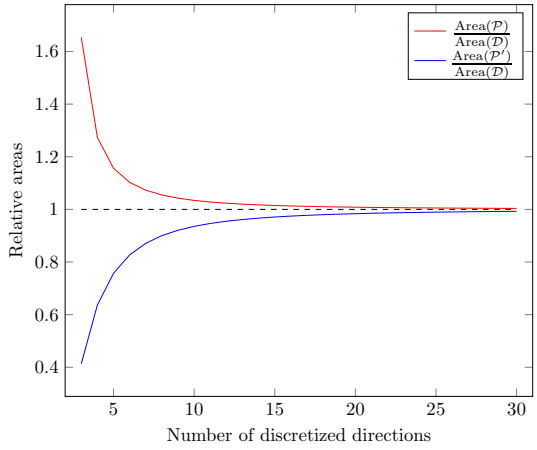
Proposition 4. *Soient $u, v \in \mathbb{R}^2$ et $d \in \mathbb{R}^+$. Nous avons l'implication suivante*

$$[\ \exists i \in \mathcal{U},\ \ \langle u - v \mid U_i \rangle \geq d\ ] \Longrightarrow ||u - v|| \geq d \tag{8.4}$$

La linéarisation de la proposition 4 fournit le polygone $\mathcal{P}$.

8.2.2 Qualité de l'approximation

Afin de comparer l'encadrement défini en 8.2.1, nous comparons les surfaces des polygones $\mathcal{P}$ et $\mathcal{P}'$ au disque unité pour plusieurs nombres de directions. Les résultats sont exposés en figure 8.2. A partir de 12 directions, la précision relative est inférieure à 3% pour la sur-approximation et inférieure à 5% pour la sous-approximation. A 20 directions, l'erreur de la sur-approximation est inférieure à 1% et celle de la sous-approximation est de 1.6%.



(a)

$n_{\text{directions}}$	$\frac{\text{Area}(\mathcal{P})}{\text{Area}(\mathcal{D})}$	$\frac{\text{Area}(\mathcal{P}')}{\text{Area}(\mathcal{D})}$
4	1.273	0.637
8	1.055	0.900
12	1.023	0.955
16	1.013	0.974
20	1.008	0.984
24	1.006	0.989
28	1.004	0.992

(b)

FIGURE 8.2 – (a-b) Évolution de l'aire des polygones approximants $\mathcal{P}$ et $\mathcal{P}'$ par rapport à l'aire d'un disque $\mathcal{D}$ de rayon unitaire.

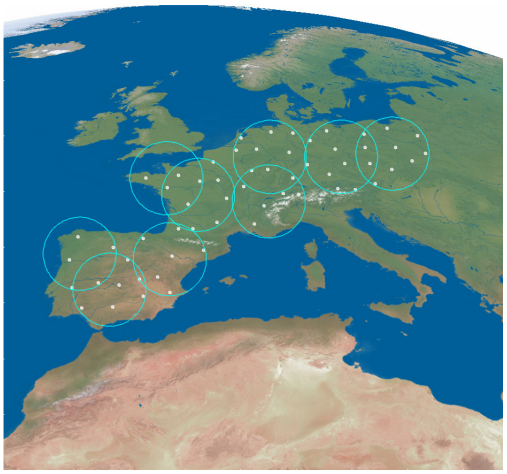
8.2.3 Utilisation pour le problème du $k-$centre continu

Nous l'illustrons l'utilisation de la linéarisation présentée précedemment avec un problème de $k-$centre continu. Ce problème, bien connu, consiste à couvrir N stations par k disques en minimisant le rayon maximum des centre. Les variables de ce problème sont les centres des disques (variables continues), le rayon maximum (continu), les variables d'allocation des stations aux disques (binaires). Ici, la proposition 3 est utilisée pour sur-estimer la valeur réelle du rayon maximum. Cette exemple est plus longuement détaillée dans [Camino 2019].

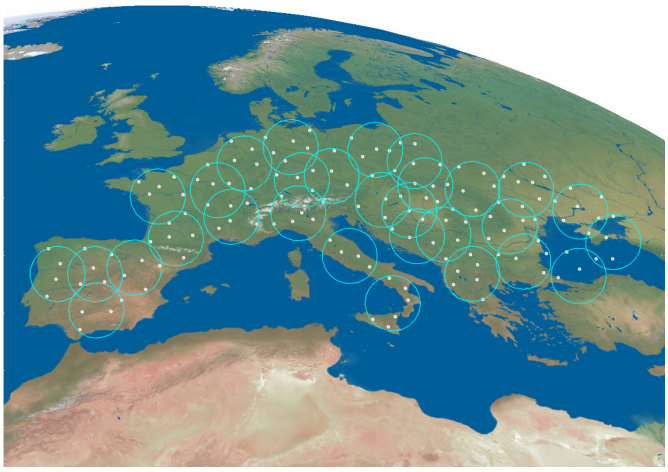
Des résultats numériques sur 3 types d'instances, 100, 120 et 130 stations, sont reportés dans le tableau 8.1. Chaque famille d'instances a été calculé avec un temps limité à 100s et pour 10 instances. Les 6 premières lignes correspondent à différente précision de notre linéarisation. Les 3 lignes suivantes correspondent à différents niveau de précision de [Liberti 2009] et les 3 dernières lignes à [Mukherjee 2013].On remarque que notre linéarisation obtient de très bons résultats et que compte-tenu du temps de calcul, un bon compromis est de considérer entre 8 et 12 directions.

8.3 Application au problème de placement de faisceaux.

Un satellite multifaisceaux est un type particulier de satellite de télécommunication qui rend un service à ses utilisateurs grâce à une pluralité de faisceaux relativement étroits, un faisceau étant une zone de puissance électromagnétique importante à la surface de la Terre pour une source radiofréquence donnée. Après avoir reçu les signaux d'une passerelle connectée au réseau terrestre, la charge utile du satellite convertit en fréquence, amplifie et retransmet les signaux d'entrée dans les différents faisceaux à travers les antennes à réflecteur, comme illustré sur la Fig. 8.4 où nous avons 13 faisceaux transmis par 4 antennes à réflecteur (une par couleur sur la figure).



(a)



(b)

FIGURE 8.3 – (a) Une solution au problème du $k-$centre avec $N = 59$, $K = 9$, $n_{\text{directions}} = 12$. (b) Une solution au problème du $k-$centre avec $N = 144$, $K = 30$, $n_{\text{directions}} = 12$

Un autre exemple présenté sur la figure 8.5 et tiré de [Camino 2016a] illustre l'intérêt du positionnement continu des faisceaux et du choix du diamètre. Sur le côté gauche avec les sous-figures (a), (c) et (e), trois dispositions de faisceau différentes et des affectations de réflecteur d'antenne représentées par les différentes couleurs sont utilisées pour couvrir la même zone de service. La disposition des faisceaux (a) est une disposition régulière telle qu'utilisée dans la pratique où 117 faisceaux sont nécessaires pour couvrir la zone. La disposition des faisceaux (c) est également une disposition régulière de 50 faisceaux identiques de diamètre, supérieur à (a). La disposition des faisceaux (e) est une disposition non régulière avec différents diamètres de sorte que seuls 48 faisceaux sont nécessaires pour couvrir la zone. Sur le côté droit, avec les sous-figures (b), (d) et (f), le trafic couvert par chaque disposition de faisceau est représenté via un autre

	$N = 100$	$N = 120$	$N = 130$
DIR4	702.1	626.3	609.7
DIR8	591.5	**556.5**	**558.4**
DIR12	**584.1**	599.5	677.3
DIR14	600.3	648.7	806.0
DIR16	596.1	700.8	785.4
DIR20	630.4	842.1	913.6
L10	699.0	746.6	779.8
L20	694.0	765.0	798.4
L30	727.9	802.5	782.5
L40	705.8	755.2	766.2
ML	597.6	733.5	765.8
MC	1116.5	1360.2	1465.0
MWC	1226.1	1480.0	1637.1

TABLE 8.1 – Valeur du rayon pour plusieurs linéarisation avec 100, 120 et 130 stations.

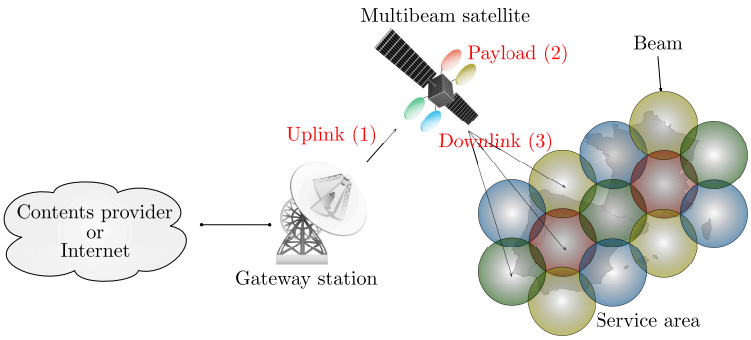
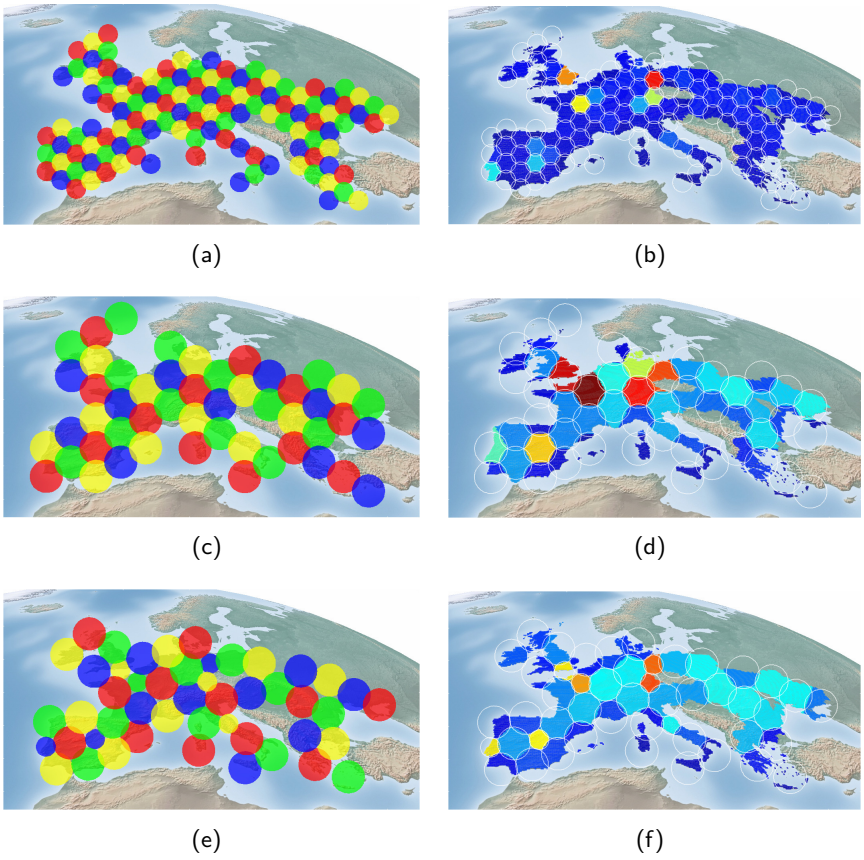


FIGURE 8.4 – Architecture standard d'un satellite de télécommunications multifaisceaux.

codage couleur. Les faisceaux bleus et bleu clair couvrent les demandes de trafic faibles et moyennes tandis que les faisceaux jaunes, orange et rouges couvrent les demandes de trafic élevées, très élevées et excessives, respectivement. La figure montre que la disposition régulière des faisceaux (a,b) est inefficace : le grand nombre de faisceaux bleus signifie que de nombreux faisceaux sont sous-utilisés et, d'autre part, il y a une demande excessive couverte par le faisceau rouge. Les dispositions de faisceaux (c,d) et (e,f) nécessitent beaucoup moins de faisceaux sous-utilisés, ce qui donne une solution plus rentable en pratique. De plus, l'utilisation de différents diamètres (e,f) évite les faisceaux surutilisés car cela permet de mieux ajuster la capacité à la demande. En fait, la disposition du faisceau (a, b) suit le soi-disant motif de réseau hexagonal avec une seule largeur de faisceau. Il s'agit d'une disposition de faisceau proche de l'optimum lorsque les demandes de trafic sont homogènes sur la zone de service, car il maximise la densité de remplissage du cercle. Cependant, l'exemple de la Fig. 8.5 montre que

ce n'est plus le cas pour des demandes de trafic hétérogènes. L'utilisation de différents diamètres de faisceau et d'une disposition de faisceau irrégulière produit de meilleures solutions.



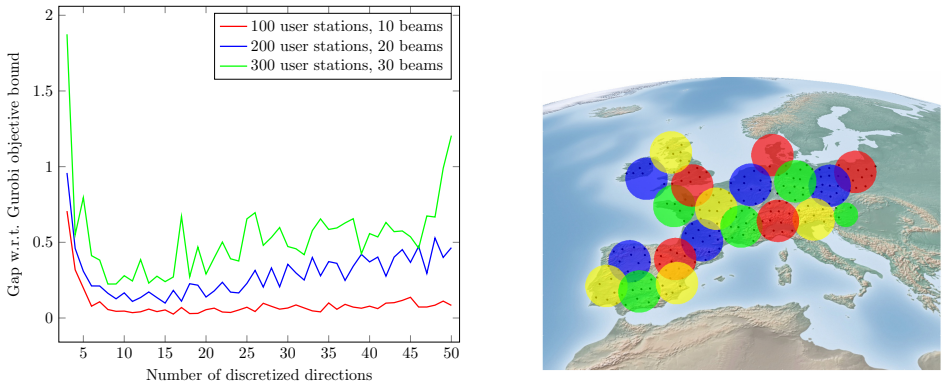
(a) (b) (c) (d) (e) (f)

FIGURE 8.5 – (a),(c),(e) Allocation de réflecteurs : 117 faisceaux réguliers, 50 faisceaux réguliers, 46 faisceaux irréguliers. (b),(d),(f) Charge par faisceau : 117 faisceaux réguliers, 50 faisceaux réguliers, 46 faisceaux irréguliers (bleu= faible, rouge= forte).

Le problème analytique n'est pas présentée en détail ici cependant, les éléments majeurs sont décrits dans les lignes suivantes. L'objectif est de servir la demande des utilisateurs. Un utilisateur ne peut être que dans un seul faisceau et un faisceau a exactement un réflecteur (une couleur). Pour qu'un faisceau couvre un utilisateur, une distance maximale doit être respectée entre l'utilisateur et le centre du faisceau, on a donc une contrainte de proximité qui est linéarisé à l'aide de la proposition 3. Toutefois, deux faisceaux avec le même réflecteur ne peuvent pas être trop proches, on a donc ici une distance minimal de séparation qui est linéarisé au moyen de de la proposition 4. Cette dernière contrainte interdit donc l'utilisation des linéarisations de [Liberti 2009] et [Mukherjee 2013] puisqu'elle n'est pas convexe.

Les résultats des expérimentations numériques sont présentés dans la figure 8.6 avec des temps de calcul limité à 500s. Il n'est pas surprenant de voir que plus il y a d'utilisateurs dans le système, plus le problème est difficile à résoudre. Ensuite, la

principale observation que l'on peut faire de ces résultats est que, pour un trop petit nombre de directions ($3 \leq n_{\text{directions}} \leq 8$), l'approximation des disques par $\mathcal{P}$ (contrainte de séparation) et $\mathcal{P}'$ (contrainte de proximité) est trop approximative et ne permet pas d'atteindre des solutions de bonne qualité. En revanche, pour un nombre trop élevé de directions ($20 \leq n_{\text{directions}} \leq 50$), le gain en précision devient si faible que les éventuelles améliorations de la qualité de la solution ne compensent pas l'augmentation de la complexité numérique due à la taille croissante du modèle : cela explique la dégradation des écarts moyens de la Fig. 8.6(a) pour ces valeurs de $n_{\text{directions}}$. En résumé pour ce problème, un bon compromis serait un nombre de directions compris entre entre 10 et 20.



(a) (b)

FIGURE 8.6 – (a) Influence du nombre de directions $n_{\text{directions}}$ sur la qualité des solutions. (b) Exemple d'une instance avec 200 utilisateurs résolue optimalement avec $n_{\text{directions}} = 12$.

8.4 Amélioration de la formulation

Plusieurs améliorations de la formulations ont été proposées dans [Camino 2021], elles visent à réduire la taille du MIP, en termes de variables binaires et de contraintes, afin d'améliorer ses performances sur de grandes instances de problèmes. Dans [Camino 2021], nous proposons un moyen de calculer une borne supérieure du nombre de faisceaux nécessaires. En effet, avoir un nombre maximum de faisceaux réduit réduit effectivement la taille du MIP. Un deuxième apport est de partitionner les stations utilisatrices en cluster et d'affecter chaque faisceau à un seul cluster, de manière à réduire les symétries et éventuellement réduire le nombre de variables et de contraintes. Nous utilisons les techniques de clustering de k-moyennes pour atteindre cet objectif. Le nombre de faisceaux par cluster est limité de deux manières. Une approche exacte préserve la généralité de la formulation en ce sens que le MIP révisé résultant conserve le même ensemble de solutions que le MIP original. Une approche heuristique effectue plus de réductions de variables et de contraintes mais donne un MIP dont la région réalisable n'est qu'un sous-ensemble de la région réalisable du MIP original, on perd

donc éventuellement l'optimalité. Des réductions significatives sont obtenues par cette dernière approche qui obtient en plus de bien meilleurs résultats que le MIP original en termes de couverture de la demande sur les grandes instances. Un exemple de clustering en k-moyennes est donné dans la figure 8.7.

Un point clé du modèle révisé réside dans le nombre maximal de faisceaux pouvant être attribués à un cluster donné. En effet, plus ce nombre est élevé, plus le nombre de variables et de contraintes du MIP révisé est important. Dans [Camino 2021], nous proposons des méthodes pour calculer une borne supérieure et la valeur exacte de ce nombre. Nous proposons également une heuristique pour diminuer davantage ce nombre, entraînant éventuellement une perte de généralité dans le sens où la solution optimale du MIP résultant peut être une solution sous-optimale du problème original.

La question du nombre de clusters est à traiter en amont de la résolution. Une idée est fournie par la figure 8.8 pour 200 utilisateurs, 2 largeurs de faisceaux, 4 réflecteurs et 12 directions discrétisés pour la linéarisation de la norme L_2. Le processus heuristique (en bleu) permet de diminuer le nombre de variables et de contraintes significativement en dessous du cas où aucun clustering n'est appliquée (en noir) alors que la méthode préservant la généralité (en rouge) échoue à apporter une réduction de variables et de contraintes, bien qu'il assure toujours la rupture de symétrie car chaque faisceau est affecté à un seul cluster. Ceci est dû au fait que plus il y a de faisceaux dans les clusters, moins le processus de réduction de la taille du modèle est efficace.

Des résultats numériques approfondis sont disponibles dans [Camino 2021], néanmoins la figure 8.9 permet d'observer brièvement une comparaison des 3 MIP pour une forte hétérogénéité de la distribution des utilisateurs sur la zone de service :

— Le modèle MIP brut sans réduction de taille (MILP1).
— Le modèle MIP basé sur un clustering en $k-$moyennes avec une preservation exacte de la généralité du problème (MILP2).
— Le modèle MIP basé sur un clustering en $k-$moyennes avec un nombre de faisceaux par cluster déterminé heuristiquement (MILP3).

La première observation est que le modèle avec clustering exact est totalement (MILP2) dominé par les autres modèles. Ensuite, on peut remarquer que plus les instances sont grandes, plus le modèle MILP2 domine. Quand les instances sont plus petites, le MILP1 reste le plus performant bien que l'écart avec MILP3 soit faible.

8.5 Conclusion

Une linéarisation originale de la norme L_2 a été proposée. Dans un plan, cette linéarisation est performante et la comparaison avec d'autres méthodes existantes est plutôt en notre faveur puisqu'elles sont, soit moins précises, soit incapable de prédire si le résultat sera une sur-approximation ou une sous-approximation (ce qui interdit toute utilisation dans un système de contraintes). On doit cependant nuancer le nombre d'applications possibles de cette linéarisation puisqu'elle n'est utilisable qu'en deux dimensions, ce qui la limite à des problèmes géométriques.

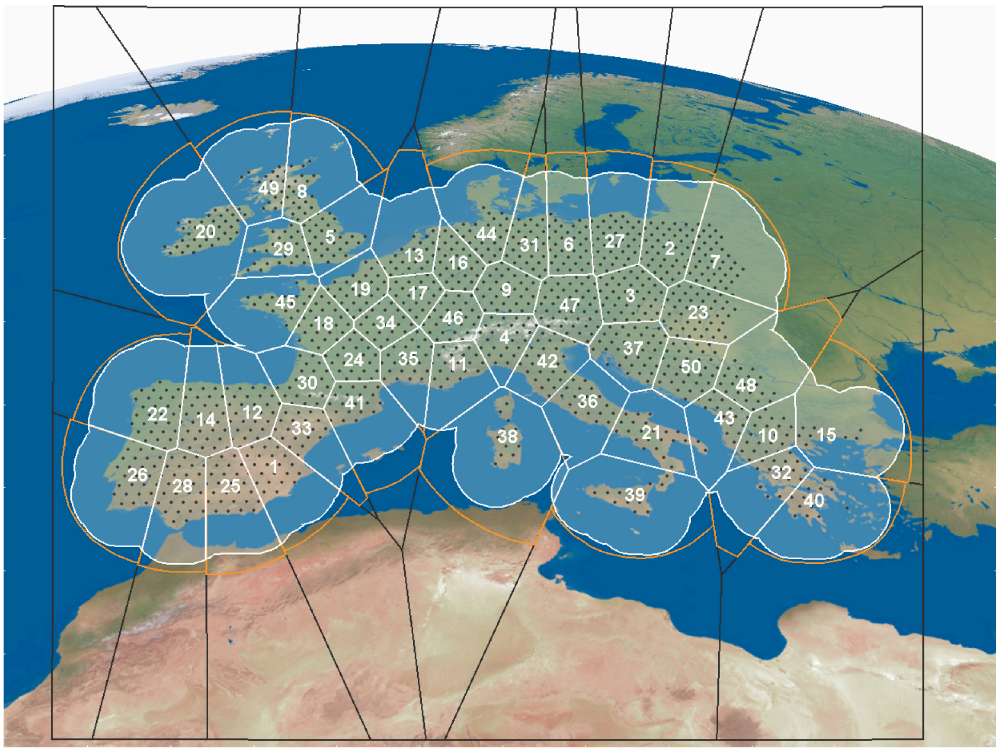


FIGURE 8.7 – Partitionnement en k-moyennes de la zone de service (k = 50) et domaines admissibles pour les centres.

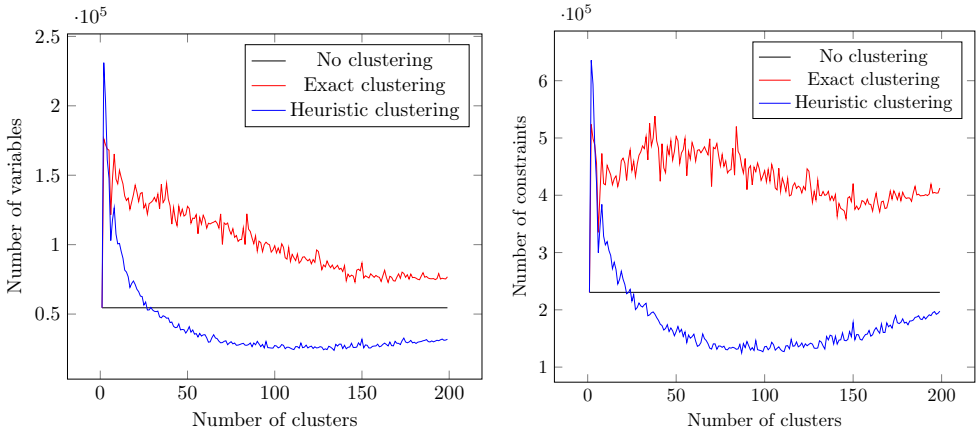
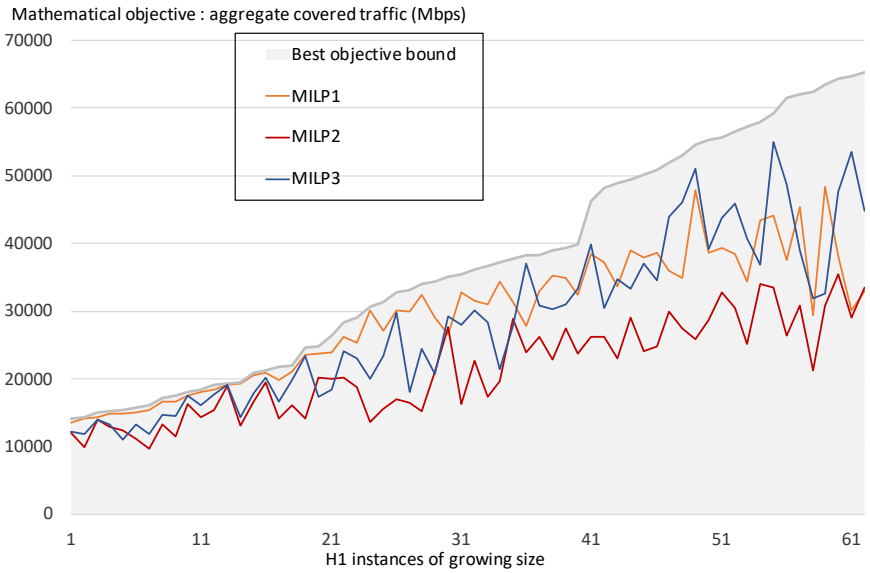


FIGURE 8.8 – Évolution du nombre de (a) variables, (b) contraintes des différents programmes linéaires suivant le nombre de clusters.



(e)

FIGURE 8.9 – Comparaison du modèle linéaire mixte de placement de faisceaux basique aux modèles avec clustering exact et heuristique.

Conclusion et perspectives

Conclusion

Dans la première partie de ce document, nous avons étudié des problèmes d'ordonnancement cyclique avec des paramètres incertains. Plus précisément, nous avons considéré le cas où les incertitudes portent sur les durées des tâches. Nous avons choisi d'utiliser une approche d'optimisation robuste. Deux problèmes ont été examinés. Le premier est le problème d'ordonnancement cyclique de base. L'objectif est alors d'ordonnancer un ensemble de tâches génériques d'une manière répétitive de sorte à minimiser le temps de cycle. Nous recherchons, plus précisément, la plus petite valeur du temps de cycle telle qu'il existe un ordonnancement pour toutes les réalisations possibles des durées de tâches dans l'ensemble d'incertitude. Le deuxième problème est celui du job shop cyclique. Ce problème correspond à un problème d'ordonnancement cyclique de base avec contraintes de ressources. L'objectif de ce problème est de trouver la plus petite valeur du temps de cycle, ainsi qu'un ordre de passage des tâches s'exécutant sur les mêmes machines de sorte qu'il existe un ordonnancement pour toutes les réalisations possibles des durées de tâches dans l'ensemble d'incertitude. Nous avons modélisé les incertitudes en considérant l'ensemble d'incertitude introduit par Bertsimas et Sim [Bertsimas 2004]. Plus précisément, chaque tâche possède une durée nominale, et une déviation par rapport à la durée nominale quand un aléas survient. De plus, un paramètre appelé budget d'incertitude limite le nombre de déviations possibles. Pour les deux problèmes d'ordonnancement considérés, nous avons utilisé une approche bi-niveaux. C'est-à-dire que certaines décisions sont prises avant de connaître la réalisation de l'incertitude et d'autres sont retardées jusqu'à la connaissance des vraies durées des tâches afin de les prendre en compte. Nous avons proposé des algorithmes efficaces adaptés aux deux problèmes étudiés. Les travaux du chapitre 5 portent sur le problème du job shop flexible. Ce problème complexe est peu étudié dans la littérature. Nous avons proposé une formulation de programmation linéaire en nombres entiers et une décomposition de Benders qui permettent de résoudre des instances ayant une taille modérée. De plus, nous avons aussi proposé deux heuristiques permettant le passage à l'échelle.

La seconde partie de ce manuscrit est consacrée à mes travaux dans le spatial. Ces travaux ont souvent été motivés par des applications industrielles avec de grandes compagnies (Airbus, Thales Alenia Space).

Le chapitre 6 met en lumière l'utilisation de l'optimisation pour l'allocation de fréquence dans les systèmes de satellite. L'objectif de ce chapitre était de quantifier le gain apporté par les techniques proposées par le SDMA (multiplication des faisceaux, dépointage). Deux modèles MIP réalistes, prenant en compte des interférences cumulatives, et non binaires, sont proposés. De plus, une seconde étape d'optimisation est proposée par des techniques de dépointage de faisceaux. Cette étude, combinant nouveauté technologique et optimisation de la ressource radio, a permis de mesurer

l'intérêt de ces nouvelles techniques. Dans le chapitre 7, nous mettons en lumière l'intérêt de l'utilisation des cliques maximales dans un problème d'affectation de bande de fréquence avec contraintes binaires. La contribution majeure du chapitre 8 est la définition d'une procédure inédite de linéarisation de contraintes inégalités impliquant des normes euclidiennes. Effectivement, nous avons démontré que, moyennant une erreur d'approximation contrôlable, nous arrivons à nous affranchir des non-linéarités ainsi que des éventuelles non-convexités des contraintes euclidiennes du problème originel de placement de faisceau. Le principe de linéarisation proposé repose sur une discrétisation des directions du plan qui permet de définir des contraintes linéaires suffisantes qui peuvent se substituer aux calculs de norme euclidienne. Le nombre de directions discrétisées est un paramètre de cette méthodologie : il permet de choisir le niveau d'approximation des distances euclidienne avec lequel nous acceptons de travailler. Ce principe de linéarisation a alors été appliqué avec succès dans une phase de modélisation du problème de placement de faisceaux dans le formalisme de la programmation linéaire mixte. La limite de cette linéarisation est son utilisation à des distances en dimension 2, ce qui la restreint à des problèmes géométriques.

Certaines activités n'ont pas été mentionnées dans ce manuscrit. Par exemple, ma collaboration avec l'équipe SARA du LAAS-CNRS sur un problème d'optimisation de la ressource radio dans les réseaux 5G dont les résultats out été publiés dans [Oussakel 2022].

Les travaux présentés dans ce document reflètent mes activités depuis ma prise de poste à l'Université de Toulouse. Cependant, durant ma thèse, j'ai eu l'occasion de m'intéresser aux systèmes dynamiques à événements discrets. Plus particulièrement, je me suis concentré sur les systèmes à événements discrets qui mettent en jeu des phénomènes de synchronisation et possèdent une description linéaire dans l'algèbre (max,+). Cette algèbre, aussi appelé algèbre des chemins, permet de facilement manipuler des graphes. Dans ces travaux, j'ai abordé des problèmes de commande des systèmes (max,+)-linéaires. Jusque-là, les commandes de systèmes (max,+)-linéaires ont principalement considéré le critère du juste-à-temps et des objectifs de commande comme la poursuite d'une trajectoire de référence ou la poursuite d'un transfert de référence. La première contribution a été la formulation d'un problème de commande pour ces systèmes en un problème de commande optimale. Ensuite, nous avons abordé un nouveau critère de commande et on s'est intéressé à la synthèse d'un correcteur qui ralentit le moins possible le transfert du système tout en assurant un objectif de commande défini par un ensemble de contraintes sur l'état. Les réseaux de transport admettent une description linéaire dans l'algèbre (max,+). Le problème de synthèse de tables d'horaires a notamment été formulé en un problème de commande de système (max,+)-linéaire. Ces travaux ont donné lieu aux deux publications suivantes : [Houssin 2007, Houssin 2013].

Perspectives

A court et à moyen terme, les perspectives de mes activités de recherches sont nombreuses. Elles se rapportent à l'ordonnancement et au milieu spatial mais aussi à d'autres champs que je souhaite explorer dans le futur. Dans la suite de ce chapitre, j'expose ces pistes, qui pour certaines, sont déjà explorées actuellement.

Ordonnancement flexible

Ce travail est actuellement en cours puisque Carla Juvin (LAAS-CNRS) a débuté sa thèse en 2020. Elle est co-encadrée par Pierre Lopez (LAAS-CNRS) et moi même.

On s'intéresse au problème d'ordonnancement de production du type job-shop flexible préemptif (*preemptive flexible job-shop scheduling problem*, pFJSSP). Trois approches sont proposées pour le résoudre : la programmation linéaire en nombres entiers (MILP), la programmation par contraintes (CP), la décomposition de Benders basée sur la logique (LBBD).

Dans le problème de job-shop (JSSP), on a un ensemble de jobs composés d'opérations et chacune d'elles doit être traitée sur une machine prédéterminée. Pour l'objectif de minimisation de la durée totale que l'on considère ici, le problème est NP-difficile. Le problème du job-shop flexible (FJSSP) est une extension du problème de job-shop classique dans lequel plusieurs machines sont en capacité d'exécuter une même opération, cette flexibilité permet alors de s'adapter aux variations du marché. Un grand nombre de méthodes, exactes et heuristiques, ont été proposées pour résoudre le FJSSP (voir [Shen 2018] pour un travail récent). Nous considérons actuellement une version préemptive de ce problème. Quand elle est possible, la préemption d'opérations en cours de traitement peut permettre de réduire les durées de fabrication. Cependant, les problèmes de job-shop préemptif ont reçu peu d'attention [Ebadi 2013], et, à notre connaissance, personne ne s'est encore intéressé à la résolution du problème de job-shop flexible et préemptif.

On propose trois méthodes exactes pour résoudre ce problème. La première méthode utilise la programmation linéaire en nombres entiers. Dans ce but, on se base sur une formulation indexée sur le temps proposée par Bowman [Bowman 1959] pour le problème de job-shop préemptif. On l'adapte pour y intégrer la notion de flexibilité dans le choix des machines.

La deuxième méthode de résolution s'appuie sur la programmation par contraintes. La formulation que nous proposons s'inspire de celle développée par [Polo Mejia 2020] pour un problème d'ordonnancement de projet, que nous avons adaptée à notre problème. Nous exploitons pour cela les fonctionnalités du solveur IBM CP Optimizer (CPO) en utilisant une décomposition d'une opération $O_{i,j}$ en $p_{i,j,m}$ parties de durée unitaire et $I_{i,j}$ une variable intervalle entre le début et la fin de traitement de l'opération.

Enfin, nous proposons de résoudre le problème avec un algorithme de décomposition de Benders basée sur la logique [Hooker 2007]. Pour cela, on décompose le problème en :

— un problème maître d'allocation des ressources résolu à l'aide de la programmation linéaire en nombres entiers et une relaxation du sous-problème ;
— un sous-problème d'ordonnancement des opérations sur les machines (job-shop préemptif) résolu soit (1) à l'aide de la programmation par contraintes, soit (2) par une version adaptée à notre problème du branch-and-bound proposé par Ebadi et Moslehi [Ebadi 2013], et générant itérativement une coupe ajoutée au problème maître.

Pour le momment, les expérimentations numériques menées sur 276 instances réparties en 8 jeux de données classiques du FJSSP [1] en 10 mn de temps de calcul ont montré : d'une part, qu'une décomposition du problème est bénéfique et permet d'obtenir de meilleurs résultats que la programmation mathématique (5% d'instances résolues à l'optimum) ou la programmation par contraintes (9%) seule ; d'autre part, que la résolution du sous-problème par le branch-and-bound est plus rapide et permet donc de résoudre plus d'instances à l'optimum (LBBD/B&B, 36%) que par programmation par contraintes (LBBD/CP, 16%). Notre travail actuel consiste à améliorer ces méthodes, en particulier la méthode de décomposition pour résoudre les instances les plus difficiles (Dauzère-Pérès&Paulli et Chambers&Barnes, les seules pour lesquelles aucune méthode ne permet une résolution optimale dans le temps imparti), et à les étendre pour d'autres fonctions objectif telles que la minimisation de la somme des dates de fin des jobs. Ces travaux ont fait l'objet de la soumission [Juvin 2022b] et de la publication [Juvin 2022a].

La version non-réemptive de ce problème est aussi à l'étude. Cependant, pour le moment les méthodes développées qui mêlent décomposition de Benders logique et relaxation lagrangienne ne sont pas encore compétitives face à CPO. Des efforts sont encore à fournir pour dépasser ce solveur si efficace sur les problèmes d'ordonnancement.

Actuellement, nous nous intéressons également à une formulation robuste du job-shop flexible dans laquelle les durées des tâches ne sont pas connues avec précision. Elle prend la forme d'une optimisation bi-niveau dans laquelle les variables de premier niveau sont l'affectation des tâches aux machines et un makespan et les variables de recours sont les séquences des tâches sur les machines ainsi que leur date de début. L'objectif est de trouver une affectation et un makespan de sorte que, quelque soit le scénario, il existe une séquence des tâches et des dates de début qui respectent le makespan. Une formulation CCG (voir 4.3.2) est en cours de développement.

Ordonnancement cyclique

Pour le moment, le MIP définit par 1.8 est très performant et les tentatives basées sur des branch-and-bound avec un branchement sur les décalages d'occurrence se sont montrées moins efficaces. Néanmoins, la porte reste ouverte à des méthodes de décomposition telle qu'une décomposition de Benders classique. En effet, la structure du problème (variables entières et variables continues) peut laisser entrevoir des résultats satisfaisants. D'autre part, une décomposition de Benders type LBBD basée sur une analyse des circuits du graphe total (disjonctif et conjonctif) peut s'avérer prometteuse.

1. http://opus.ub.hsu-hh.de/volltexte/2012/2982/ – Dernier accès octobre 2021

TABLE 8.2 – Exemple de flowshop sans attente.

Job	Machine	Temps de set-up	Durée
J_1	M_1	2	3
	M_2	3	2
	M_3	1	2
J_2	M_1	3	2
	M_2	3	$1 \rightarrow 3$
	M_3	4	3
J_3	M_1	3	2
	M_2	1	3
	M_3	2	3

En effet, une analyse du circuit critique du graphe fournit une coupe logique minimale et donc plus globale. Il existe des algorithmes polynomiaux pour trouver ces circuits critiques.

Flowshop robuste sans attente

Plus récemment, je me suis intéressé à ce problème avec Ron McGarvey (University of Missouri). Ici, on se concentre sur l'optimisation du makespan pour un problème de flowshop sans attente avec des temps de préparation (temps de *set-up*) et des temps de traitement incertains. On formule le problème comme un problème robuste bi-niveau dans lequel la séquence des tâches sur les machines est fixée ainsi que le makespan lors du premier niveau et les variables de recours regroupent des dates de début des tâches. Classiquement, le problème maître fixe une séquence et un makespan, puis le sous-problème a pour but de trouver un scénario, dans les limites fixées par un budget d'incertitude, qui viole le makespan. Ainsi, ce scénario est ajouté au problème maître et on itère jusqu'à ce que le sous-problème ne trouve plus de scénario qui viole le makespan. Comme souvent dans ce type de problème, il est difficile de formuler le sous-problème efficacement. Pour le moment, notre sous-problème est un MIP. A noter que l'originalité de ce problème réside dans les contraintes *no-wait* qui peuvent conduire à une réduction du makespan quand la durée d'une tâche augmente, ce qui est peu intuitif.

Un exemple est présenté dans le tableau 8.2. La figure 8.10 décrit l'ordonnancement sans attente $J_1 \rightarrow J_2 \rightarrow J_3$, le makespan est alors de 23 u.t.. Si la durée de la seconde tâche du job 2 passe de 1 à 3 u.t. alors le makespan du même ordonnancement décroît pour devenir égal à 21. Cela s'explique par la caractéristique "sans attente" du problème d'ordonnancement. Dans ces conditions, l'ordonnancement est reproduit en figure 8.11.

Maintenance prédictive : priorisation des réparations

Les sous-traitants de services d'assistance pour les équipements de grande valeur, tels que les avions, soutiennent de plus en plus des stratégies de contrats basés sur la performance (PBC), dans le cadre desquelles les contrats sont conçus pour optimiser la préparation du système. L'une des décisions clés auxquelles un tel entrepreneur est

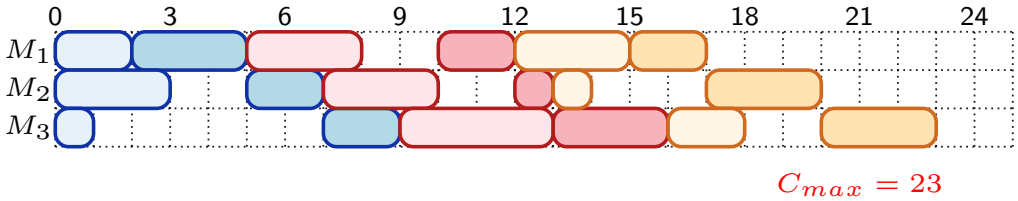


FIGURE 8.10 – L'ordonnancement sans attente $J_1 \rightarrow J_2 \rightarrow J_3$.

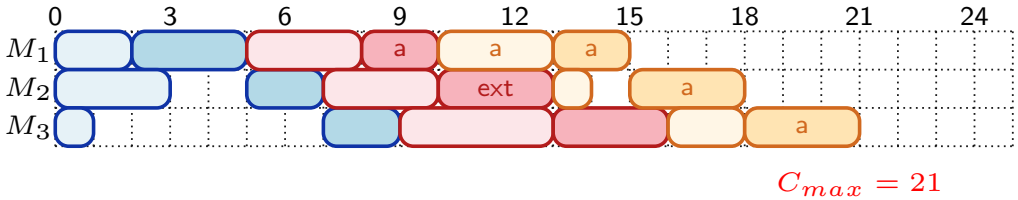


FIGURE 8.11 – L'ordonnancement sans attente $J_1 \rightarrow J_2 \rightarrow J_3$ avec une tâche de durée prolongée. Un "a" à l'intérieur d'une tâche signifie que la tâche a été avancée.

confronté est de déterminer, en temps réel, quels composants défaillants doivent être réparés, étant donné un ensemble fini de ressources. L'impact de toute défaillance ou réparation de composant individuel peut avoir un effet sur la disponibilité des avions. En supposant qu'une fois la réparation lancée, elle ne peut pas être interrompue (et doit donc s'exécuter jusqu'à la fin), il peut être préférable de différer la réparation d'un composant de faible priorité défaillant, dans le cas où un composant de priorité supérieure défaillant pourrait nécessiter une réparation dans l'avenir. Par exemple, cela peut s'appliquer dans les cas où la réparation du composant de faible priorité nécessite une longue durée.

Les principaux défis analytiques sont

— la défaillance future des composants est inconnue,
— le temps de réparation réel des composants est inconnu,
— la "priorité" relative d'un composant donné, en termes d'effet sur la disponibilité de l'avion, est dynamique et ne peut pas être facilement spécifié *a priori*.

Une approche pour répondre à ces incertitudes consiste à modéliser l'entreprise de réparation comme un système stochastique. En effet, le problème peut se modéliser sous la forme d'un processus de Markov à temps discret. L'objectif est alors de trouver la politique qui minimise un critère constitué des coûts d'immobilisation d'un avion, de la pénurie des pièces et de l'écart du stock de pièces de rechange par rapport à une valeur de consigne.

Ces travaux, qui viennent de débuter, sont le fruit d'une collaboration avec Ron McGarvey (University of Missouri) et Alain Haït (ISAE-Supaero).

Optimisation de la couverture partielle dans les réseaux

La conception de réseaux est un large domaine pour lequel les modèles dépendent des applications désirées. Une classification de ces problèmes est disponible dans [Magnanti 1984] et un point commun de ces problèmes est que leur modélisation conduit à des problèmes d'optimisation discrets ou mixtes (arbre couvrant, plus court chemin, emplacement d'installation...). Les champs d'application concernent les transports, les télécommunications, l'énergie, les chaînes logistiques. Ces problèmes trouvent également naturellement leur place dans le tournant écologique que doit prendre le 21e siècle : conception des réseaux électriques, détermination des transports publics, gestion du trafic et de la congestion en environnement urbains.

Il est généralement trop coûteux de connecter tous les nœuds d'un réseau, c'est pourquoi un choix judicieux doit être fait pour déterminer un sous-réseau qui satisfait la demande. Ce sous-réseau peut être spécifié par un ensemble de paires origine-destination. Parmi les problèmes les plus connus de conception de réseaux, le problème de la couverture maximale sous des contraintes de budget (MC) et celui de la couverture partielle sous des contraintes de réalisation minimale (PC) présentent un grand intérêt dans de nombreux champs d'application. Les deux problèmes considèrent un ensemble de paires origine-destination pondérées par un trafic entre les deux sommets.

Le problème MC est assez classique dans la littérature, il consiste à couvrir au mieux la demande des paires origine-destination tout en respectant un budget alloué à la construction du réseau. Concernant le problème PC, la problématique a été peu étudiée à part récemment dans [Bucarey 2022]. Dans ce dernier problème, on cherche à minimiser le coût du réseau sous des contraintes de service minimum (plus précisément, un seuil de demande à respecter). Dans ([Bucarey 2022]), les auteurs ont proposé une méthode basée sur la décomposition de Benders pour la résolution de ces 2 problèmes.

La prise en compte des incertitudes qui peuvent affecter les données de ces problèmes conduit à développer de nouveaux modèles d'optimisation et à adapter les techniques de résolution. Plus précisément, la version robuste du problème MC a été étudiée et récemment [Conde 2021] ont proposé une décomposition de Benders pour le problème de minimisation du regret. Cependant, il n'y a pas, à notre connaissance, de version robuste du problème (PC).

Une perspective de mes travaux de recherche est de proposer de nouveaux algorithmes de résolution basés sur des outils modernes de programmation mathématique entière tels que la décomposition de Benders, la génération de colonnes ou la relaxation lagrangienne pour la résolution de problèmes de conception de réseaux dans leur version nominal ou robuste. Dans un premier temps la version robuste du problème PC peut être étudiée. Les versions « min regret » et « minimax » présentent toutes les deux un intérêt. D'autre part, l'étude des versions de ce problème avec capacité sur les nœuds peut également être envisagée.

Un sujet de thèse a été déposé sur cette thématique avec Sonia Cafieri (ENAC).

Ordonnancement des contacts Search And Rescue dans une constellation de satellites de géopositionnement

Le *Search And Rescue* (SAR) est un service de localisation et d'aide aux personnes en détresse. Il est implémenté dans plusieurs constellations de satellites de géolocalisation récentes telles que GPS, GALILEO ou GLONASS. Cependant, la mise en place de ce service est conditionnée par la bonne gestion des contacts entre les utilisateurs et les satellites d'une part et entre les satellites et les stations (passerelles) au sol d'autre part. La planification des contacts SAR peut être considérée comme un double problème d'affectation avec fenêtres de temps. Plus précisément, il faut associer à chaque utilisateur au moins un et au mieux deux satellites visibles (orbite basse, non géostationnaire) et que ces satellites soient associés à une antenne d'une station au sol (visible par le satellite). L'objectif est alors de définir un ordonnancement des affectations dans le temps tel que, à chaque instant, chaque utilisateur est associé idéalement à deux satellites qui sont eux-mêmes associés à une antenne dans leurs fenêtres de visibilité.

Dans le problème considéré, la constellation est composée de 18, 24 ou 32 satellites suivant les instances. Les sites qui rassemblent les antennes (un site comporte 4 antennes) sont au nombre de 5 et ils sont situés aux endroits suivants : Nouméa, Kourou, Svalbard, Tahiti, Réunion. L'objectif de ce problème est de fournir une affectation des antennes aux satellites sur un horizon de 10,5 jours en tenant compte des fenêtres de visibilités des satellites par rapport aux sites en maximisant le taux de couverture moyen des utilisateurs. L'algorithme utilisé actuellement pour la gestion du SAR dans la constellation GALILEO est celui présenté dans [Porretta 2018]. C'est un algorithme glouton qui sélectionne en premier lieu l'affectation des satellites aux sites puis dans un second temps, une affectation plus fine qui correspond à l'association des satellites aux antennes. Cet algorithme a un taux de couverture satisfaisant en pratique mais peut générer beaucoup de handovers (changements d'antennes intempestifs pour un satellite). L'idée, ici, est d'améliorer la résolution du problème via une approche de programmation mathématique.

Un modèle de PLNE a déjà été développé visant à pénaliser les handovers. Le PLNE consiste à affecter des antennes aux satellites sur un intervalle de temps où les visibilités antennes/satellites et utilisateurs/satellites ne changent pas. Des instances ont été générées à partir des indications disponibles dans l'article [Porretta 2018] et des données publiques. Le taux de couverture moyen du modèle MIP est quasiment identique à celui de l'algorithme de l'ESA, soit 99,5% en moyenne (pourcentage du temps où chaque utilisateur est couvert par deux satellites). Cependant le nombre de handovers est nettement inférieur pour notre modèle (546 handovers pour le PLNE contre 23047 pour l'algorithme de l'ESA) ce qui est plus intéressant d'un point de vue pratique. La figure 8.12 illustre le taux de couverture des utilisateurs. Des méthodes de décompositions sont actuellement envisagées pour améliorer encore les résultats et accélérer les temps de calcul. Ce travail est commun avec Christian Artigues (LAAS-CNRS).

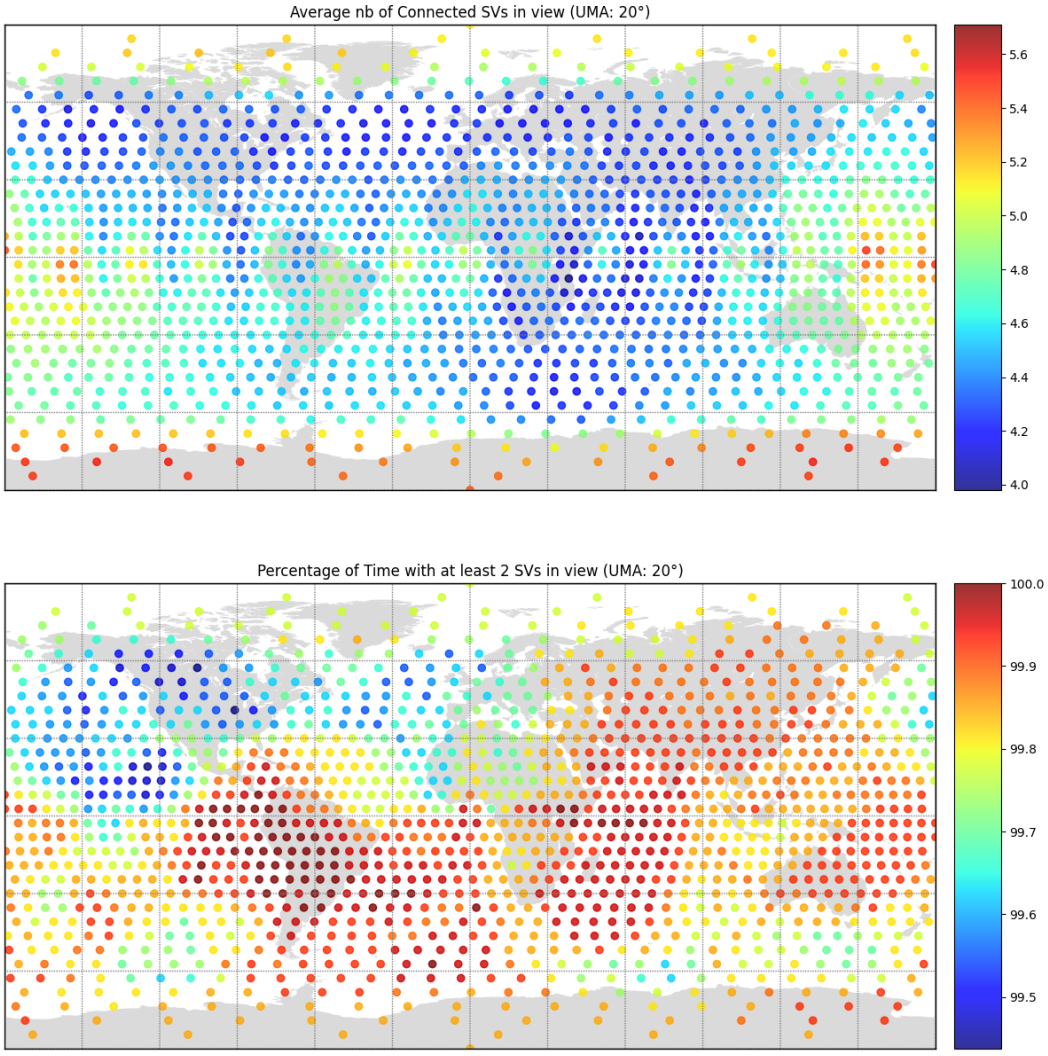


FIGURE 8.12 – Taux de couverture sur un horizon de 80000s.

Problème du *Circle Packing*

Récemment, les auteurs de [Gleixner 2020] ont proposé une formulation en MINLP (Mixed Integer Nonlinear Programming : programmation entière non -linéaire) puis une méthode de génération de colonnes pour le problème de *Circle Packing*. La génération de colonne semble efficace mais le sous-problème reste non-linéaire. En effet, le problème de *pricing* a pour but de générer de nouveaux motifs mais il est NP-difficile et compliqué à résoudre en pratique même pour un faible nombre de cercles. Une linéarisation de la norme L_2 comme celle présentée dans le §8.2.1 pourrait aider à résoudre ce sous-problème puisqu'il fait appel à des contraintes de séparation et des contraintes de proximité.

Optimisation discrète et machine learning

Il est difficile d'occulter l'intérêt des techniques d'apprentissage pour l'optimisation. Plusieurs travaux, tous publiés récemment, vont dans ce sens. On peut, par exemple, mentionner les travaux de [Zhou 2016] qui permettent de prédire comment initialiser une recherche locale ou les travaux de [Kruber 2017] qui prédisent si une méthode de génération de colonnes peut être utile à une instance donnée. Concernant l'ordonnancement, nous pouvons citer les travaux récents de [Parmentier 2021] sur les problèmes d'ordonnancement à une seule machine. L'optimisation robuste a aussi bénéficié de ces nouvelles techniques dans [Kämmerling 2019]. Plus précisément, les auteurs prédisent un sous-ensemble de scénarios pertinents à incorporer au problème maître afin de réduire le nombre d'itérations avec le sous-problème.

Pour ma part, je pense que ces techniques permettraient de prédire des coupes moins exclusives et plus globales dans le cas d'une utilisation de la LBBD. En effet, la coupe remontée par le sous-problème dans un schéma LBBD est souvent peu efficiente car trop restrictive. Aussi, générer une coupe mettant en jeu un plus petit nombre de variables du problème maître pourrait grandement accélérer ce type de schéma. L'écueil de cette technique est qu'il faut que la coupe reste valide pour le problème maître, ce qui impose de résoudre le sous-problème.

Bibliographie

[Aardal 2003] K. I. Aardal, C. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino et A. Sassano. *Models and Solution Techniques for the Frequency Assignment Problem*. 4OR, vol. 1, no. 4, pages 261–317, 2003. (Cité en page 77.)

[Aissi 2009] Hassene Aissi, Cristina Bazgan et Daniel Vanderpooten. *Min–max and min–max regret versions of combinatorial optimization problems : A survey*. European journal of operational research, vol. 197, no. 2, pages 427–438, 2009. (Cité en page 24.)

[Ales 2018] Zacharie Ales, Thi Sang Nguyen et Michael Poss. *Minimizing the weighted sum of completion times under processing time uncertainty*. Electronic Notes in Discrete Mathematics, vol. 64, pages 15 – 24, 2018. 8th International Network Optimization Conference - INOC 2017. (Cité en page 34.)

[Artigues 2013] Christian Artigues, Roel Leus et Fabrice Talla Nobibon. *Robust optimization for resource-constrained project scheduling with uncertain activity durations*. Flexible Services and Manufacturing Journal, vol. 25, no. 1, pages 175–205, Jun 2013. (Cité en page 34.)

[Atamturk 2007] Alper Atamturk et Muhong Zhang. *Two-stage robust network flow and design under demand uncertainty*. Operations Research, vol. 55, no. 4, pages 662–673, 2007. (Cité en page 31.)

[Ayoub 2016] Josette Ayoub et Michael Poss. *Decomposition for adjustable robust linear optimization subject to uncertainty polytope*. Computational Management Science, vol. 13, no. 2, pages 219–239, Apr 2016. (Cité en pages 32 et 53.)

[Azzouz 2017] Ameni Azzouz, Meriem Ennigrou et Lamjed Ben Said. *A hybrid algorithm for flexible job-shop scheduling problem with setup times*. International Journal of Production Management and Engineering, vol. 5, no. 1, 2017. (Cité en page 62.)

[Balouka 2018] Noemie Balouka et Izack Cohen. *A Robust Optimization Model for the Multi-mode Resource Constrained Project Scheduling Problem with Uncertain Activity Durations*. Dans International Conference on Project Management and Scheduling (PMS), page 4p., Roma, Italy, avril 2018. (Cité en page 34.)

[Barman 2018] J. M. Barman, C. Martinez et S. Verma. *Max-plus to solve the cyclic job shop problem with time lags*. Dans 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), pages 785–790, 2018. (Cité en page 11.)

[Ben Hmida 2010] Abir Ben Hmida, Mohamed Haouari, Marie-José Huguet et Pierre Lopez. *Discrepancy Search for the Flexible Job Shop Scheduling Problem*. Computers and Operations Research, vol. 37, no. 12, pages p. 2192–2201, décembre 2010. 27 pages. (Cité en page 62.)

[Ben-Tal 1999] A. Ben-Tal et A. Nemirovski. *Robust solutions of uncertain linear programs*. Operations Research Letters, vol. 25, no. 1, pages 1 – 13, 1999. (Cité en pages 26, 28 et 29.)

[Ben-Tal 2000a] Aharon Ben-Tal et Arkadi Nemirovski. *Robust solutions of linear programming problems contaminated with uncertain data*. Mathematical programming, vol. 88, no. 3, pages 411–424, 2000. (Cité en pages 23 et 26.)

[Ben-Tal 2000b] Aharon Ben-Tal et Arkadi Nemirovski. *Robust solutions of Linear Programming problems contaminated with uncertain data*. Mathematical Programming, vol. 88, no. 3, pages 411–424, Sep 2000. (Cité en pages 27 et 30.)

[Ben-Tal 2002] Aharon Ben-Tal et Arkadi Nemirovski. *Robust optimization–methodology and applications*. Mathematical Programming, vol. 92, no. 3, pages 453–480, 2002. (Cité en page 23.)

[Ben-Tal 2004] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer et Arkadi Nemirovski. *Adjustable robust solutions of uncertain linear programs*. Mathematical Programming, vol. 99, no. 2, pages 351–376, 2004. (Cité en page 30.)

[Ben-Tal 2009] Aharon Ben-Tal, Laurent El Ghaoui et Arkadi Nemirovski. Robust optimization, volume 28. Princeton University Press, 2009. (Cité en pages 23 et 25.)

[Benabid-Najjar 2012] Abir Benabid-Najjar, Claire Hanen, Olivier Marchetti et Alix Munier-Kordon. *Periodic schedules for bounded timed weighted event graphs*. IEEE Transactions on Automatic Control, vol. 57, no. 5, pages 1222–1232, 2012. (Cité en page 11.)

[Benabid 2011] Abir Benabid et Claire Hanen. *Worst case analysis of decomposed software pipelining for cyclic unitary RCPSP with precedence delays*. Journal of Scheduling, vol. 14, no. 5, pages 511–522, Oct 2011. (Cité en page 8.)

[Bendotti 2017] Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux et Alain Quilliot. *Anchored reactive and proactive solutions to the CPM-scheduling problem*. European Journal of Operational Research, vol. 261, no. 1, pages 67 – 74, 2017. (Cité en page 35.)

[Bertsimas 2003] Dimitris Bertsimas et Melvyn Sim. *Robust discrete optimization and network flows*. Mathematical programming, vol. 98, no. 1-3, pages 49–71, 2003. (Cité en page 30.)

[Bertsimas 2004] Dimitris Bertsimas et Melvyn Sim. *The Price of Robustness*. Operations Research, vol. 52, no. 1, pages 35–53, 2004. (Cité en pages 26, 27, 37, 38, 47 et 103.)

[Bertsimas 2009] Dimitris Bertsimas et David B Brown. *Constructing uncertainty sets for robust linear optimization*. Operations research, vol. 57, no. 6, pages 1483–1495, 2009. (Cité en page 24.)

[Bertsimas 2016] Dimitris Bertsimas, Iain Dunning et Miles Lubin. *Reformulation versus cutting-planes for robust optimization*. Computational Management Science, vol. 13, no. 2, pages 195–217, 2016. (Cité en page 26.)

[Birge 2011] John R Birge et Francois Louveaux. Introduction to stochastic programming. Springer Science & Business Media, 2011. (Cité en page 23.)

[Bouchard 2010] Mathieu Bouchard, Mirjana Čangalović et Alain Hertz. *On a reduction of the interval coloring problem to a series of bandwidth coloring problems*. Journal of Scheduling, vol. 13, pages 583–595, 12 2010. (Cité en page 85.)

[Bougeret 2018] Marin Bougeret, Artur Alves Pessoa et Michael Poss. *Robust scheduling with budgeted uncertainty*. Discrete Applied Mathematics, 2018. (Cité en page 33.)

[Bouman 2011] Paul C Bouman, JM Van Den Akker et JA Hoogeveen. *Recoverable robustness by column generation*. Dans European Symposium on Algorithms, pages 215–226. Springer, 2011. (Cité en page 31.)

[Boussemart 2002] Frédéric Boussemart, Guillaume Cavory et Christophe Lecoutre. *Solving the cyclic job shop scheduling problem with linear precedence constraints using CP techniques*. Dans Systems, Man and Cybernetics, 2002 IEEE International Conference on, volume 7, pages 6–pp. IEEE, 2002. (Cité en page 20.)

[Bowman 1959] Edward H. Bowman. *The Schedule-Sequencing Problem*. Operations Research, vol. 7, pages 621–624, 1959. (Cité en page 105.)

[Bowman 1993] R. A. Bowman et J. A. Muckstadt. *Stochastic Analysis of Cyclic Schedules*. Operations Research, vol. 41, no. 5, pages 947–958, 1993. (Cité en page 33.)

[Bożejko 2015] Wojciech Bożejko, Jarosław Pempera et Mieczysław Wodecki. *Parallel Simulated Annealing Algorithm for Cyclic Flexible Job Shop Scheduling Problem*. Dans Leszek Rutkowski, Marcin Korytkowski, Rafal Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh et Jacek M. Zurada, editeurs, Artificial Intelligence and Soft Computing, 2015. (Cité en page 62.)

[Brélaz 1979] D. Brélaz. *New methods to color the vertices of a graph*. Communications of the ACM, vol. 22, pages 251–256, 1979. (Cité en page 77.)

[Bron 1973] Coen Bron et Joep Kerbosch. *Algorithm 457 : Finding All Cliques of an Undirected Graph*. Commun. ACM, vol. 16, no. 9, page 575–577, sep 1973. (Cité en page 87.)

[Brucker 1990] P. Brucker et R. Schlie. *Job-shop scheduling with multi-purpose machines*. Computing, vol. 45, no. 4, pages 369–375, Dec 1990. (Cité en page 62.)

[Bruni 2017] M.E. Bruni, L. Di Puglia Pugliese, P. Beraldi et F. Guerriero. *An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations*. Omega, vol. 71, pages 66 – 84, 2017. (Cité en page 34.)

[Bruni 2018] M.E. Bruni, L. Di Puglia Pugliese, P. Beraldi et F. Guerriero. *A computational study of exact approaches for the adjustable robust resource-constrained project scheduling problem*. Computers& Operations Research, vol. 99, pages 178 – 190, 2018. (Cité en page 34.)

[Bucarey 2022] Víctor Bucarey, Bernard Fortz, Natividad González-Blanco, Martine Labbé et Juan A. Mesa. *Benders decomposition for network design covering problems*. Computers & Operations Research, vol. 137, page 105417, 2022. (Cité en page 109.)

[Buchheim 2017] Christoph Buchheim et Jannis Kurtz. *Robust combinatorial optimization under convex and discrete cost uncertainty*. Rapport technique, Technical report, Optimization Online, 2017. (Cité en page 30.)

[Camino 2014] Jean-Thomas Camino, Stéphane Mourgues, Christian Artigues et Laurent Houssin. *A greedy approach combined with graph coloring for non-uniform beam layouts under antenna constraints in multibeam satellite systems*. Dans 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2014. (Cité en page 92.)

[Camino 2015] Jean-Thomas Camino, Christian Artigues, Laurent Houssin et Stéphane Mourgues. *A Decomposition Method for Frequency Assignment in Multibeam Satellite Systems*. Dans Begoña Vitoriano et Greg H. Parlier, editeurs, ICORES 2015 - Proceedings of the International Conference on Operations Research and Enterprise Systems, Lisbon, Portugal, 10-12 January, 2015, pages 23–33. SciTePress, 2015. (Cité en page 92.)

[Camino 2016a] Jean-Thomas Camino, Christian Artigues, Laurent Houssin et Stéphane Mourgues. *Mixed-integer linear programming for multibeam satellite systems design : Application to the beam layout optimization*. Dans Annual IEEE Systems Conference, SysCon 2016, Orlando, FL, USA, April 18-21, 2016, pages 1–6. IEEE, 2016. (Cité en page 95.)

[Camino 2016b] Jean-Thomas Camino, Christian Artigues, Laurent Houssin et Stéphane Mourgues. *Mixed-integer linear programming for multibeam satellite systems design : Application to the beam layout optimization*. Dans 2016 Annual IEEE Systems Conference (SysCon), pages 1–6, 2016. (Cité en page 92.)

[Camino 2019] Jean-Thomas Camino, Christian Artigues, Laurent Houssin et Stéphane Mourgues. *Linearization of Euclidean norm dependent inequalities applied to multibeam satellites design*. Comput. Optim. Appl., vol. 73, no. 2, pages 679–705, 2019. (Cité en pages 92 et 94.)

[Camino 2021] Jean-Thomas Camino, Christian Artigues, Laurent Houssin et Stéphane Mourgues. *MILP formulation improvement with* k*-means clustering for the beam layout optimization in multibeam satellite systems*. Comput. Ind. Eng., vol. 158, 2021. (Cité en pages 92, 98 et 99.)

[Carlier 1978] Jacques Carlier. *Ordonnancements a contraintes disjonctives*. RAIRO-Operations Research, vol. 12, no. 4, pages 333–350, 1978. (Cité en page 15.)

[Cavory 2005] Guillaume Cavory, Remy Dupas et Gilles Goncalves. *A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints*. European Journal of Operational Research, vol. 161, no. 1, pages 73–85, 2005. (Cité en page 20.)

[Celebi 2011] M. Celebi, F. Celiker et H. Kingravi. *On Euclidean Norm Approximations*. Pattern Recognition, pages 278–283, 2011. (Cité en page 92.)

[Chauvet 2003] Fabrice Chauvet, Jeffrey W Herrmann et J-M Proth. *Optimization of cyclic production systems : a heuristic approach*. IEEE Transactions on Robotics and Automation, vol. 19, no. 1, pages 150–154, 2003. (Cité en page 16.)

[Che 2015] Ada Che, Jianguang Feng, Haoxun Chen et Chengbin Chu. *Robust optimization for the cyclic hoist scheduling problem*. European Journal of Operational Research, vol. 240, no. 3, pages 627 – 636, 2015. (Cité en page 33.)

[Chen 1999] H. Chen, J. Ihlow et C. Lehmann. *A genetic algorithm for flexible job-shop scheduling*. Dans Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), volume 2, pages 1120–1125 vol.2, 1999. (Cité en page 62.)

[Chrétienne 1984] P Chrétienne. *Chemins extrémaux d'un graphe doublement valué.* RAIRO, Rech. Opér., vol. 18, pages 221–245, 1984. (Cité en page 14.)

[Chretienne 1991] Philippe Chretienne. *The basic cyclic scheduling problem with deadlines*. Discrete Applied Mathematics, vol. 30, no. 2, pages 109 – 123, 1991. (Cité en page 14.)

[Cochet-Terrasson 1998] Jean Cochet-Terrasson, Guy Cohen, Stéphane Gaubert, Michael McGettrick et Jean-Pierre Quadrat. *Numerical computation of spectral elements in max-plus algebra*. Dans Proc. IFAC Conf. on Syst. Structure and Control, 1998. (Cité en pages 14 et 43.)

[Conde 2021] Eduardo Conde et Marina Leal. *A robust optimization model for distribution network design under a mixed integer set of scenarios*. Computers & Operations Research, vol. 136, page 105493, 2021. (Cité en page 109.)

[Cuninghame-Green 1962] Raymond A Cuninghame-Green. *Describing industrial processes with interference and approximating their steady-state behaviour*. Journal of the Operational Research Society, vol. 13, no. 1, pages 95–100, 1962. (Cité en page 11.)

[D'Ambrosio 2017] C. D'Ambrosio et L. Liberti. *Distance Geometry in Linearizable Norms*. Dans Geometric Science of Information, pages 830–837. Springer International Publishing, 2017. (Cité en page 92.)

[Dantzig 1955] George B. Dantzig. *Linear Programming under Uncertainty*. Management Science, vol. 1, no. 3/4, pages 197–206, 1955. (Cité en page 23.)

[Dasdan 1998] Ali Dasdan, S Irani et Rajesh K Gupta. *An experimental study of minimum mean cycle algorithms*. Rapport technique, Tech. rep. 98-32, Univ. of California, Irvine, 1998. (Cité en page 14.)

[Dasdan 1999] Ali Dasdan, Sandy S Irani et Rajesh K Gupta. *Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems*. Dans Design Automation Conference, 1999. Proceedings. 36th, pages 37–42. IEEE, 1999. (Cité en page 14.)

[Dasdan 2004] Ali Dasdan. *Experimental analysis of the fastest optimum cycle ratio and mean algorithms*. ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 9, no. 4, pages 385–418, 2004. (Cité en pages 14 et 43.)

[Dawande 2005] Milind Dawande, H. Neil Geismar, Suresh P. Sethi et Chelliah Sriskandarajah. *Sequencing and Scheduling in Robotic Cells : Recent Developments*. Journal of Scheduling, vol. 8, no. 5, pages 387–426, Oct 2005. (Cité en page 8.)

[de Werra 1994] D. de Werra et Y. Gay. *Chromatic scheduling and frequency assignment*. Discrete Applied Mathematics, vol. 49, no. 1, pages 165–174, 1994. Special Volume Viewpoints on Optimization. (Cité en page 85.)

[Duguet 2022] Aloïs Duguet, Christian Artigues, Laurent Houssin et Sandra Ulrich Ngueveu. *Properties, Extensions and Application of Piecewise Linearization for Euclidean Norm Optimization in* $\mathbb{R}^2$. Journal of Optimization Theory and Applications, vol. 195, no. 2, pages 418–448, 2022. (Cité en page 92.)

[Dunkin 1998] N. W. Dunkin, J. E. Bater, P. G. Jeavons et D. A. Cohen. *Towards High Order Constraint Representations for the Frequency Assignment Problem*. Dans CSD-TR-98-05, Royal Holloway, University of London, Egham, Surrey, UK, 1998. (Cité en page 77.)

[Ebadi 2013] Abbas Ebadi et Ghasem Moslehi. *An optimal method for the preemptive job shop scheduling problem*. Computers & Operations Research, vol. 40, no. 5, pages 1314–1327, 2013. (Cité en pages 105 et 106.)

[El Ghaoui 1997] L. El Ghaoui et H. Lebret. *Robust Solutions to Least-Squares Problems with Uncertain Data*. SIAM Journal on Matrix Analysis and Applications, vol. 18, no. 4, pages 1035–1064, 1997. (Cité en pages 26 et 29.)

[El Ghaoui 1998] L. El Ghaoui, F. Oustry et H. Lebret. *Robust Solutions to Uncertain Semidefinite Programs*. SIAM Journal on Optimization, vol. 9, no. 1, pages 33–52, 1998. (Cité en pages 26 et 29.)

[Fink 2012] Martin Fink, Touria Ben Rahhou et Laurent Houssin. *A new procedure for the cyclic job shop problem*. IFAC Proceedings Volumes, vol. 45, no. 6, pages 69–74, 2012. (Cité en pages 19, 20 et 53.)

[Gasperoni Asperoni 1994] Franco Gasperoni Asperoni et Uwe Schwiegelshohn. *Generating close to optimum loop schedules on parallel processors*. Parallel Processing Letters, vol. 04, no. 04, pages 391–403, 1994. (Cité en page 8.)

[Ghadiri Nejad 2018] Mazyar Ghadiri Nejad, Gergely Kovács, Béla Vizvári et Reza Vatankhah Barenji. *An optimization model for cyclic scheduling problem in flexible robotic cells*. The International Journal of Advanced Manufacturing Technology, vol. 95, no. 9, pages 3863–3873, Apr 2018. (Cité en page 62.)

[Gleixner 2020] Ambros Gleixner, Stephen Maher, Benjamin Müller et João Pedro Pedroso. *Price-and-verify : a new algorithm for recursive circle packing using Dantzig-Wolfe decomposition*. Annals of Operations Research, vol. 284, no. 2, pages 527 – 555, 2020. (Cité en page 111.)

[Gondran 1979] M. Gondran et M. Minoux. Graphes et algorithmes. Eyrolles, Paris, 1979. Engl. transl. *Graphs and Algorithms*, Wiley, 1984. (Cité en page 14.)

[Graham 2008] J. Graham, Roberto Montemanni, J. Moon et D. Smith. *Frequency assignment, multiple interference and binary constraints*. Wireless Networks, vol. 14, pages 449–464, 08 2008. (Cité en page 85.)

[Hamaz 2018a] Idir Hamaz, Laurent Houssin et Sonia Cafieri. *A Branch-and-Bound Procedure for the Robust Cyclic Job Shop Problem*. Dans Combinatorial Optimization - 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers, volume 10856 of *Lecture Notes in Computer Science*, pages 228–240. Springer, 2018. (Cité en pages 43 et 45.)

[Hamaz 2018b] Idir Hamaz, Laurent Houssin et Sonia Cafieri. *A robust basic cyclic scheduling problem*. EURO Journal on Computational Optimization, vol. 6, no. 3, pages 291–313, 2018. (Cité en pages 43 et 45.)

[Hamaz 2022] Idir Hamaz, Laurent Houssin et Sonia Cafieri. The robust cyclic job shop problem. Submitted, mai 2022. (Cité en page 56.)

[Hanen 1994] Claire Hanen. *Study of a NP-hard cyclic scheduling problem : The recurrent job-shop*. European journal of operational research, vol. 72, no. 1, pages 82–101, 1994. (Cité en pages 11, 13, 15, 18, 20 et 53.)

[Hanen 1997] Claire and Hanen. Journal of the operational research society, chapitre Cyclic scheduling on parallel processors : An overview, pages 194–226. Taylor & Francis, Oxford, 1997. (Cité en pages 8, 14 et 18.)

[Hanzalek 2011] Zdenek Hanzalek, Claire Hanen et Premysl Sucha. *Cyclic Scheduling-New Application and Concept of Core Precedences*. eraerts, page 165, 2011. (Cité en page 8.)

[Herroelen 2005] Willy Herroelen et Roel Leus. *Project scheduling under uncertainty : Survey and research potentials*. European journal of operational research, vol. 165, no. 2, pages 289–306, 2005. (Cité en page 35.)

[HITZ 1979] KL HITZ. *Scheduling of Flexible Flow Shops I*. Tech. Rep. LIDS-R-879, MIT, Cambridge, MA, 1979. (Cité en page 8.)

[Hooker 2007] John Hooker. *Planning and Scheduling by Logic-Based Benders Decomposition*. Operations Research, vol. 55, pages 588–602, 06 2007. (Cité en page 105.)

[Houssin 2007] Laurent Houssin, Sébastien Lahaye et Jean-Louis Boimond. *Just in Time Control of Constrained (* max *, +)-Linear Systems*. Discret. Event Dyn. Syst., vol. 17, no. 2, pages 159–178, 2007. (Cité en page 104.)

[Houssin 2011a] Laurent Houssin. *Cyclic jobshop problem and (max,plus) algebra*. Dans World IFAC Congress (IFAC 2011), pages 2717–2721, Milan, Italy, août 2011. (Cité en page 11.)

[Houssin 2011b] Laurent Houssin, Christian Artigues et Erwan Corbel. *Frequency allocation problem in a SDMA satellite communication system*. Comput. Ind. Eng., vol. 61, no. 2, pages 346–351, 2011. (Cité en pages 74 et 77.)

[Houssin 2013] Laurent Houssin, Sébastien Lahaye et Jean-Louis Boimond. *Control of (max, +)-linear systems minimizing delays*. Discret. Event Dyn. Syst., vol. 23, no. 3, pages 261–276, 2013. (Cité en page 104.)

[Howard 1964] Ronald A Howard. *Dynamic programming and Markov processes*. 1964. (Cité en pages 14 et 43.)

[Hsu 2002] Tiente Hsu, R. Dupas et G. Goncalves. *A genetic algorithm to solving the problem of flexible manufacturing system cyclic scheduling*. Dans IEEE International Conference on Systems, Man and Cybernetics, volume 3, 2002. (Cité en page 62.)

[Jalilvand-Nejad 2013] Amir Jalilvand-Nejad et Parviz Fattahi. *A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem*. vol. 26, 10 2013. (Cité en page 62.)

[Juvin 2022a] Carla Juvin, Laurent Houssin et Pierre Lopez. *Logic-based Benders Decomposition for preemptive Flexible Job-Shop Scheduling*. Dans 18th International Conference on Project Management and Scheduling (PMS 2022), Ghent, Belgium, avril 2022. (Cité en page 106.)

[Juvin 2022b] Carla Juvin, Laurent Houssin et Pierre Lopez. Logic-based Benders decomposition for the preemptive Flexible Job-Shop Scheduling Problem. working paper or preprint, juin 2022. (Cité en page 106.)

[Karabati 1998] S. Karabati et B. Tan. *Stochastic Cyclic Scheduling Problem in Synchronous Assembly and Production Lines*. The Journal of the Operational Research Society, vol. 49, no. 11, pages 1173–1187, 1998. (Cité en page 33.)

[Kats 2002] Vladimir Kats et Eugene Levner. *Cyclic scheduling in a robotic production line*. Journal of Scheduling, vol. 5, no. 1, pages 23–41, 2002. (Cité en page 8.)

[Kechadi 2013] M-Tahar Kechadi, Kok Seng Low et G. Goncalves. *Recurrent neural network approach for cyclic job shop scheduling problem*. Journal of Manufacturing Systems, vol. 32, no. 4, pages 689 – 699, 2013. (Cité en pages 20 et 62.)

[Kelley 1960] James E Kelley Jr. *The cutting-plane method for solving convex programs*. Journal of the society for Industrial and Applied Mathematics, vol. 8, no. 4, pages 703–712, 1960. (Cité en page 32.)

[Kiatmanaroj 2012a] Kata Kiatmanaroj, Christian Artigues, Laurent Houssin et Erwan Corbel. *Greedy algorithms for time-frequency allocation in a SDMA satellite communication system*. Dans 9th International Conference on Modeling, Optimization & SIMulation, Bordeaux, France, juin 2012. (Cité en page 74.)

[Kiatmanaroj 2012b] Kata Kiatmanaroj, Christian Artigues, Laurent Houssin et Frédéric Messine. *Frequency allocation in a SDMA satellite communication system with beam moving*. Dans Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012, pages 3265–3269. IEEE, 2012. (Cité en page 74.)

[Kiatmanaroj 2013] Kata Kiatmanaroj, Christian Artigues, Laurent Houssin et Frédéric Messine. *Frequency assignment in a SDMA satellite communication system with beam decentring feature*. Comput. Optim. Appl., vol. 56, no. 2, pages 439–455, 2013. (Cité en pages 74 et 80.)

[Kiatmanaroj 2016] Kata Kiatmanaroj, Christian Artigues et Laurent Houssin. *On scheduling models for the frequency interval assignment problem with cumulative interferences*. Engineering Optimization, vol. 48, no. 5, pages pp.740–755, mai 2016. (Cité en page 86.)

[Kouvelis 2013] Panos Kouvelis et Gang Yu. Robust discrete optimization and its applications, volume 14. Springer Science & Business Media, 2013. (Cité en page 24.)

[Kruber 2017] Markus Kruber, Marco E. Lübbecke et Axel Parmentier. *Learning When to Use a Decomposition*. Dans Domenico Salvagnin et Michele Lombardi, editeurs, Integration of AI and OR Techniques in Constraint Programming. Springer International Publishing, 2017. (Cité en page 112.)

[Kämmerling 2019] Nicolas Kämmerling et Jannis Kurtz. *Oracle-Based Algorithms for Binary Two-Stage Robust Optimization*, 2019. (Cité en page 112.)

[LEV 2022] *Exact solutions for the two-machine robust flow shop with budgeted uncertainty*. European Journal of Operational Research, vol. 300, no. 1, pages 46–57, 2022. (Cité en page 34.)

[Levner 2007] Eugene Levner, Vladimir Kats et David Alcaide López De Pablo. *Cyclic scheduling in robotic cells : an extension of basic models in machine scheduling theory*. Dans Multiprocessor scheduling, theory and applications. InTech, 2007. (Cité en page 8.)

[Liberti 2009] L. Liberti, N. Maculan et Y. Zhang. *Optimal configuration of gamma ray machine radiosurgery units : the sphere covering subproblem*. Optimization Letters, vol. v. 3, n. 1, pages 109–121, 2009. (Cité en pages 92, 94 et 97.)

[Liebchen 2007] Christian Liebchen. *Periodic timetable optimization in public transport*. Dans Operations Research Proceedings 2006, pages 29–36. Springer, 2007. (Cité en page 8.)

[Magnanti 1981] T.L. Magnanti et Richard Wong. *Accelerating Benders Decomposition : Algorithmic Enhancement and Model Selection Criteria*. Operations Research, vol. 29, 06 1981. (Cité en page 65.)

[Magnanti 1984] T.L. Magnanti et Richard Wong. *Network Design and Transportation Planning : Models and Algorithms*. Transportation Science, vol. 18, no. 1, pages 1–55, 1984. (Cité en page 109.)

[Mannino 2003] C. Mannino et A. Sassano. *An Enumerative Algorithm for the Frequency Assignment Problem*. Discrete Applied Mathematics, vol. 129, no. 1, pages 155–169, 2003. (Cité en page 77.)

[Minoux 2009] Michel Minoux. Robust linear programming with right-hand-side uncertainty, duality and applicationsrobust linear programming with right-hand-side uncertainty, duality and applications, pages 3317–3327. Springer US, Boston, MA, 2009. (Cité en pages 32, 34 et 40.)

[Montemanni 2003] R. Montemanni, J.N.J. Moon et D.H. Smith. *An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem*. IEEE Transactions on Vehicular Technology, vol. 52, no. 4, pages 891–901, 2003. (Cité en page 77.)

[Mukherjee 2013] J. Mukherjee. *Linear combination of norms in improving approximation of Euclidean norm*. Pattern Recognition Letters, pages 1348–1355, 2013. (Cité en pages 92, 94 et 97.)

[Murphey 1999] R.A. Murphey, P. M. Pardalos et M. G. C. Resende. Handbook of combinatorial optimization : Supplement volume a, chapitre Frequency Assignment Problems. Springer, 1999. (Cité en page 77.)

[Nouri 2018] Houssem Eddine Nouri, Olfa Belkahla Driss et Khaled Ghédira. *Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model*. Journal of Industrial Engineering International, vol. 14, no. 1, Mar 2018. (Cité en page 62.)

[Oussakel 2022] Imane Oussakel, Philippe Owezarski, Pascal Berthou et Laurent Houssin. *Toward Radio Access Network Slicing Enforcement in Multi-cell 5G System*. Journal of Network and Systems Management, vol. 31, no. 1, page 8, 2022. (Cité en page 104.)

[Palpant 2008] M. Palpant, C. Oliva, C. Artigues, P. Michelon et M. Didi Biha. *Models and methods for frequency allocation with cumulative interference constraints*. International Transactions in Operational Research, vol. 15, no. 3, pages 307–324, 2008. (Cité en page 77.)

[Parmentier 2021] Axel Parmentier et Vincent T'Kindt. *Learning to solve the single machine scheduling problem with release times and sum of completion times*, 2021. (Cité en page 112.)

[Pinedo 2008] Michael Pinedo. Scheduling : Theory, algorithms, and systems. 01 2008. (Cité en page 85.)

[Polo Mejia 2020] Oliver Polo Mejia, Christian Artigues, Pierre Lopez et Virginie Basini. *Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility*. International Journal of Production Research, vol. 58, no. 23, pages 7149–7166, 2020. (Cité en page 105.)

[Porretta 2018] Marco Porretta, Bernhard Kleine Schlarmann, Alexandre Ballereau et Massimo Crisci. *A Novel Uplink Scheduling Algorithm for the Galileo System*. IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 2, pages 819–833, 2018. (Cité en page 110.)

[Poss 2014] Michael Poss. *Robust combinatorial optimization with variable cost uncertainty*. European Journal of Operational Research, vol. 237, no. 3, pages 836 – 845, 2014. (Cité en page 30.)

[Quezada 2020] Franco Quezada, Céline Gicquel, Safia Kedad-Sidhoum et Dong Quan Vu. *A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales*. Computers and Operations Research, vol. 116, page 104865, avril 2020. (Cité en page 31.)

[Quinton 2018] Felix Quinton, Idir Hamaz et Laurent Houssin. *Algorithms for the flexible cyclic jobshop problem*. Dans 14th IEEE International Conference on Automation Science and Engineering, CASE 2018, Munich, Germany, August 20-24, 2018, pages 945–950. IEEE, 2018. (Cité en page 62.)

[Quinton 2020] Felix Quinton, Idir Hamaz et Laurent Houssin. *A mixed integer linear programming modelling for the flexible cyclic jobshop problem*. Ann. Oper. Res., vol. 285, no. 1, pages 335–352, 2020. (Cité en pages 62 et 64.)

[Resende 2006] M. G. C. Resende et P. M. Pardalos. Handbook of optimization in telecommunications. Springer, 2006. (Cité en page 77.)

[Romanovskil 1967] IV Romanovskil. *Optimization of stationary control of a discrete deterministic process*. Cybernetics, vol. 3, no. 2, pages 52–62, 1967. (Cité en page 14.)

[Rossi 2016] André Rossi, Evgeny Gurevsky, Olga Battaïa et Alexandre Dolgui. *Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations*. Discrete Applied Mathematics, vol. 208, pages 123 – 136, 2016. (Cité en page 35.)

[Roundy 1992] Robin Roundy. *Cyclic schedules for job shops with identical jobs*. Mathematics of operations research, vol. 17, no. 4, pages 842–865, 1992. (Cité en page 20.)

[Sadeh 1993] Norman Sadeh, Shinichi Otsuka et Robert Schnelbach. *Predictive and reactive scheduling with the Micro-Boss production scheduling and control system*. Dans Proceedings, IJCAI-93 Workshop on Knowledge-Based Production Planning, Scheduling and Control, 1993. (Cité en page 34.)

[Santos 2018] Marcio Costa Santos, Michael Poss et Dritan Nace. *A perfect information lower bound for robust lot-sizing problems*. Annals of Operations Research, Jun 2018. (Cité en page 31.)

[Shen 2018] Liji Shen, Stéphane Dauzère-Pérès et Janis S. Neufeld. *Solving the flexible job shop scheduling problem with sequence-dependent setup times*. European Journal of Operational Research, vol. 265, no. 2, pages 503–516, 2018. (Cité en page 105.)

[Singh 2014] Manjeet Singh et Robert P. Judd. *Efficient calculation of the makespan for job-shop systems without recirculation using max-plus algebra*. International Journal of Production Research, vol. 52, no. 19, pages 5880–5894, 2014. (Cité en page 11.)

[Song 1998] Ju-Seog Song et Tae-Eog Lee. *Petri net modeling and scheduling for cyclic job shops with blocking*. Computers & Industrial Engineering, vol. 34, no. 2, pages 281 – 295, 1998. (Cité en page 11.)

[Soyster 1973] A. L. Soyster. *Technical Note—Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming*. Operations Research, vol. 21, no. 5, pages 1154–1157, 1973. (Cité en page 27.)

[Sucha 2008] Premysl Sucha et Zdenek Hanzalek. *Deadline constrained cyclic scheduling on pipelined dedicated processors considering multiprocessor tasks and changeover times*. Mathematical and Computer Modelling, vol. 47, no. 9-10, pages 925–942, 2008. (Cité en page 8.)

[Thiele 2009] Aurélie Thiele, Tara Terry et Marina Epelman. *Robust linear optimization with recourse*. Rapport technique, pages 4–37, 2009. (Cité en page 32.)

[Yanikoglu 2017] Ihsan Yanikoglu, B Gorissen et Dick den Hertog. *Adjustable Robust Optimization–A Survey and Tutorial*. Available online at ResearchGate, 2017. (Cité en page 31.)

[Yu 2017] Lianfei Yu, Cheng Zhu, Jianmai Shi et Weiming Zhang. *An Extended Flexible Job Shop Scheduling Model for Flight Deck Scheduling with Priority, Parallel Operations, and Sequence Flexibility*. Sci. Program., vol. 2017, février 2017. (Cité en page 62.)

[Zhang 1997] Hongtao Zhang et Stephen C Graves. *Cyclic scheduling in a stochastic environment*. Operations Research, vol. 45, no. 6, pages 894–903, 1997. (Cité en page 33.)

[Zhang 2015] Hongchang Zhang, Simon Collart-Dutilleul et Khaled Mesghouni. *Cyclic Scheduling of Flexible Job-shop with Time Window Constraints and Resource Capacity Constraints*. vol. 48, pages 816–821, 12 2015. (Cité en page 62.)

[Zhou 2016] Yangming Zhou, Jin-Kao Hao et Béatrice Duval. *Reinforcement learning based local search for grouping problems : A case study on graph coloring*. Expert Systems with Applications, vol. 64, pages 412–422, 2016. (Cité en page 112.)

Résumé :

L'optimisation de processus complexes fait l'objet d'études en Recherche Opérationnelle et Optimisation Mathématique. Mes travaux en optimisation se sont concentrés sur deux types d'application : les problèmes d'ordonnancement et les problèmes issus du spatial. Parmi les problèmes d'ordonnancement, les problèmes cycliques correspondent à ceux pour lesquelles les tâches se répètent périodiquement. Ces problèmes ont été étudiés dans la littérature mais la plupart des travaux considèrent des paramètres déterministes. Pourtant, des incertitudes, comme la durée d'execution des tâches, peuvent survenir. Mes travaux sur l'ordonnancement cyclique visent à considérer ces incertitudes sous la forme d'un problème d'optimisation robuste bi-niveau. Une méthode de résolution basée sur une décomposition de Benders pour la version flexible du problème d'ordonnancement cyclique constitue une autre contribution dans ce domaine. Concernant les problématiques du spatial, les technologies modernes posent de nouveaux problèmes d'optimisation que nous tentons de résoudre tels que l'optimisation du placement de faisceau d'un satellite de télécommunication. Pour résoudre ce problème, nous proposons un encadrement paramétrable de la norme euclidienne dans le plan.

Mots clés : ordonnancement cyclique, optimisation robuste, optimisation combinatoire

Abstract : Several studies on cyclic scheduling problems have been presented in the literature. However, most of them consider that the problem parameters are deterministic and do not consider possible uncertainties on these parameters. However, the best solution for a deterministic problem can quickly become the worst one in the presence of uncertainties, involving bad schedules or infeasibilities. Many sources of uncertainty can be encountered in scheduling problems, for example, activity durations can decrease or increase, machines can break down, new activities can be incorporated, etc. In this PhD thesis, we focus on scheduling problems that are cyclic and where activity durations are affected by uncertainties. More precisely, we consider an uncertainty set where each task duration belongs to an interval, and the number of parameters that can deviate from their nominal values is bounded by a parameter called budget of uncertainty. This parameter allows us to control the degree of conservatism of the resulting schedule. In particular, we study two cyclic scheduling problems. The first one is the basic cyclic scheduling problem (BCSP). We formulate the problem as a two-stage robust optimization problem and, using the properties of this formulation, we propose three algorithms to solve it. The second considered problem is the cyclic jobshop problem (CJSP). As for the BCSP, we formulate the problem as two-stage robust optimization problem and by exploiting the algorithms proposed for the robust BCSP we propose a Branch-and-Bound algorithm to solve it. In order to evaluate the efficiency of our method, we compared it with classical decomposition methods for two-stage robust optimization problems that exist in the literature. Numerical experiments assess the efficiency of the proposed methods.

Keywords : cyclic scheduling, robust optimization, combinatorial optimization