N° d'ordre : 41/2023.D/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté de Mathématiques

THESE DE DOCTORAT EN SCIENCES

Présentée pour l'obtention du grade de DOCTEUR

En : MATHÉMATIQUES

Spécialité : Recherche Opérationnelle

Par : BAAZIZ Adlane

Sujet

Sur des problèmes de multicoloration de graphes et des problèmes d'ordonnancement d'atelier de production

Soutenue publiquement, le 08/10/2023, devant le jury composé de :

M. MOULAÏ Mustapha	Professeur à l'USTHB	Président
M. AIT HADDADENE Hacène	Professeur à l'USTHB	Directeur de thèse
M. OULAMARA Ammar	Professeur à l'Université de Lorraine, Nancy	Co-Directeur de thèse
M. AIDER Meziane	Professeur à l'USTHB	Examinateur
M. CHELLALI Mustapha	Professeur à l'université de Blida	Examinateur
M. IKHLEF-ECHOUF Noureddine	Professeur, Université de Médéa	Examinateur

Remerciments

En premier lieu, je remercie dieu, le tout puissant de m'avoir donné le privilège et la chance d'étudier, qui m'a guidé dans la voie de la lumière et de la science et du savoir pour réaliser cette thèse.

Je tiens à exprimer ma reconnaissance à mon directeur de thèse Monsieur AIT HADDADENE Hacène et à mon Co-directeur de thèse, Monsieur OULAMARA Ammar, qui m'ont dirigé tout au long de cette thèse, leurs conseils ainsi que les nombreuses discussions enrichissantes ont été précieux dans la réalisation et l'aboutissement de ce travail de recherche.

Je tiens à adresser mes sincères salutations à l'ensemble du jury pour avoir acepter d'évaluer ce travail : le président, Monsieur M. MOULAÏ , les examinateurs, Monsieur M. AIDER, Monsieur M. CHELLALI, ainsi que Monsieur N. IKHLEF-ECHOUF.

Je voudrais remercier particulièrement Monsieur KOUIDER Ahmed pour son aide précieuse, sa gentillesse, sa bonne humeur, le temps qui m'a consacré, les nombreux encouragements qu'il m'a prodigués.

Je remercie les membres de la faculté de Mathématiques l'USTHB, et en particulier les enseignants de la recherche opérationnelle et les membres du laboratoire LaROMaD.

Merci à tous mes amis qui m'ont soutenu au cours de ces dernières années.

Merci à tous,
Adlane BAAZIZ

Dédicase

Je dédie cette thèse qui est le fruit de tout un long chemin d'études :

Au plus beau cadeau que le bon dieu nous a offert, ceux que je due à leurs faveurs tous ce que je suis maintenant, ceux qui m'ont aidé d'achever mon chemin d'affranchir la vie, ceux qui ont toujours été là pour moi, mes très chers parents.

A ma très chère mère, envers qui je ne pourrais jamais solder la dette indéfinie que je le suis due. Que dieu la garde pour moi.

A mon très cher père, il m'a été d'une aide précieuse et dont le sérieux et la discipline ont été et seront toujours pour moi un exemple à suivre.

A ma femme, qui a joué un très grand rôle dans ce travail en me donnant un environnement idéal, et pour ses encouragements permanents.

A mes chères soeurs, mon frère Sofiane, mes deux très chers enfants Mohamed Safouane, Youcef Anes et mon neveu Ilyes, je leur souhaite un avenir plein de joie, de bonheur et de succès.

A tous mes amis pour leur soutien tout au long de mon parcours, Boukhalfa Zahout, Salah Eddine Messekher, Lyes Ghemit, Mohamed Achour, Mourad Allalou, pour tous les moments qu'on a passés ensemble.

A ceux qui ont pris une place dans mon cœur et je n'ai pas cité bien sûr ne croyez pas que je vous ai oublié je vous porte toujours dans mon cœur.

Résumé

Les travaux de recherche abordés dans cette thèse portent sur des domaines très connus de la recherche opérationnelle qui sont l'ordonnancement d'ateliers de production à objectifs multiples et la coloration de graphes. Pour l'ordonnancement, il s'agit d'organiser l'exécution d'un ensemble de tâches préemptives qui doivent être traités, avant une échéance globale prédéfinie, sur un ensemble de ressources. Chaque tâche possède son propre temps de traitement et un gain prédéterminé qui lui est attribué lorsqu'il est complètement exécuté. Les ressources se distinguent en deux types : les ressources disjonctives et les ressources cumulatives. Les jobs nécessitant la même ressource critique sont soumis à des contraintes conflictuelles interdisant leur exécution en même temps.

Ce problème classé NP-difficile (Thevenin et al., 2017), est réduit à un problème de multicoloration de graphes défini par un graphe simple non orienté G = (V, E), où V est l'ensemble des sommets et E est l'ensemble des arêtes. Chaque sommet v_i représente un job j_i, et il existe une arête (v_i, v_j) entre deux sommets v_i et v_j si les deux jobs correspondants sont incompatibles, c'est-à-dire nécessitant une même ressource critique, et par conséquent les sommets nécessitent des vecteurs de couleurs disjoints.

Dans un premier temps, nous nous sommes intéressés à la résolution du problème mono-objectif issu de l'agrégation des objectifs considérés. Pour ce faire, une approche de recuit simulé **ISA** est proposée. Cette approche est dotée d'un mécanisme de refroidissement modifié, ainsi qu'une technique adaptée de coloration de sommets. Les expérimentations sur des benchmarks de la littérature ont été réalisées. Les expérimentations ont montré que l'ap-

proche est efficace et possède une bonne performance relativement aux résultats d'autres méthodes de la littérature.

Dans un second temps, nous avons considéré le problème de multicoloration de graphes avec deux objectifs distincts à optimiser, à savoir : (i) la maximisation du gain total des sommets entièrement colorés et (ii) la minimisation de la somme de la plus grande valeur des couleurs affectées aux sommets. Pour cela, nous avons proposé une approche de recherche taboue multi-objectif, **BITS** qui détermine pour chaque solution du voisinage un ensemble de solutions non dominées. Les expérimentations sur des benchmarks ont attesté la supériorité des résultats obtenus comparativement aux résultats de la méthode NSGA-II.

Mots clés : *ordonnancement, multicoloration de graphes, optimisation multi-objectif.*

Abstract

In this thesis we address to well-known problems of operations research, namely the production scheduling and the graph multi-coloring. The goal of scheduling problem is to organize the execution of a set of preemptive jobs that must be processed, before a predefined global deadline, on a set of resources. Each job has its own processing time and a predetermined gain assigned to it when it is completely executed. The resources are divided into two types : disjunctive resources and cumulative resources. Jobs requiring the same disjunctive resource are subject to conflicting constraints prohibiting their execution at the same time.

This NP-hard problem (Thevenin et al., 2017) is modeled as a graph multi-coloring problem defined by an simple undirected graph G = (V, E), where V is the set of vertices and E is the set of edges. Each vertex v_i represents a job j_i , and there exists an edge (v_i, v_j) between two vertices v_i and v_j if the two corresponding jobs are incompatible, i.e., require the same disjunctive resource, and therefore the vertices require disjoint color vectors.

First, we interested in solving the single-objective problem resulting from the aggregation of the considered objectives. For this purpose, a simulated annealing approach **ISA** is proposed. This approach features a modified cooling mechanism, as well as an adapted vertex coloring technique. Experiments on literature benchmarks have been performed. The experiments have shown that the approach is efficient and has a good performance compared to the results of other methods in the literature.

Then, we interested to non-dominated solutions with two distinct objectives to be optimized, namely : (i) maximizing the total gain of fully colored ver-

tices and (ii) minimizing the sum of the largest value of the colors assigned to the vertices. For this purpose, we proposed a bi-objective tabu search approach **BITS**, which determines for each solution in the neighborhood a set of non-dominated solutions. Experiments on benchmarks have attested the superiority of the results The experiments on benchmarks have confirmed the superiority of the results obtained compared to the results of the **NSGA-II** method.

Keywords : *scheduling, graph multicoloring, multi-objective optimization, Meta-heuristic approach.*

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Introduction générale

De nombreux problèmes rencontrés dans différents secteurs économiques, scientifiques et techniques, sont de nature combinatoire. Selon le nombre de critères pris en compte, ces problèmes peuvent être mono-critère ou multi-critères. En effet, l'optimisation combinatoire regroupe une large classe de problèmes ayant des applications dans de nombreux domaines de l'industrie et les services, à savoir, les finances, le transport, les télécommunications, la planification et l'ordonnancement, etc. Particulièrement, au cours de cette thèse, nous nous sommes intéressés aux problèmes d'ordonnancement, ces problèmes sont présents dans le secteur manufacturier et constituent une fonction importante en gestion de production.

Les problèmes d'ordonnancement s'intéressent à l'allocation des ressources aux tâches sur des périodes données de temps à déterminer, et dont l'objectif est d'optimiser un ou plusieurs critères, ces ressources et ces tâches peuvent prendre différentes formes en pratique. Les ressources peuvent être des machines, des ouvriers ou tout simplement des outils de fabrication dans un atelier de production. Elles peuvent être aussi des unités de calcul ou des serveurs dans un environnement informatique. Les tâches sont des opérations sur des produits à fabriquer sur les machines dans un système de production. Elles peuvent être des programmes à exécuter ou simplement des requêtes à transmettre sur des serveurs dans un environnement informatique. Quelque soit le domaine d'application, l'ordonnancement contribue à l'élaboration de meilleures stratégies à adopter pour l'optimisation et l'utilisation des ressources.

La coloration de graphes est l'une des branche importante de la théorie des graphes. Elle consiste à attribuer une couleur à chaque sommet du graphe de manière que deux sommets adjacents soient de couleurs différentes. La coloration de graphes offre une large possibilité de modéliser de nombreux problèmes d'optimisation combinatoire, notamment la planification des horaires, la conception des réseaux de télécommunication, l'optimisation des ressources, ainsi que les problèmes d'ordonnancement lorsque certaines tâches sont soumises à des exigences d'incompatibilité, elles sont alors assignées à des classes de couleurs différents. Dans certains cas l'attribution d'une seule couleur à chacun des sommets du graphe présente des limites de modélisation des problèmes pratique. En effet, la multicoloration de graphes est une extention à la coloration qui attribue non seulement une couleur mais un ensemble de couleur à chacun des sommets du graphe. Cette extention est proposée dans la littérature pour étendre la possibilité de modéliser des problèmes pratiques avec des contraintes plus complexes necessitant l'attribution de plus d'une couleur à un sommet.

Dans cette thèse, nous abordons un problème particulier d'ordonnancement à l'aide de la multicoloration de graphes, dont les principales contributions se résument comme suit :

- Proposition d'une stratégie de coloration des sommets d'un graphe de conflit modélisant le problème d'ordonnancement étudié.
- Proposition d'une approche de recuit simulé basée sur un mécanisme de refroidissement modifié pour la résolution du problème de multicoloration de graphes.
- Proposition d'une approche de recherche tabou bi-objectif pour la multicoloration de graphes.

C'est dans cette optique que les travaux de cette thèse sont réalisés et décrits dans ce manuscrit qui est composé principalement de quatre chapitres organisés comme suit :

- Le premier chapitre constitue une introduction générale aux problèmes

d'ordonnancement dans la gestion de production. Il présente diverses notions essentielles pour la compréhension de l'ensemble des travaux présentées dans cette thèse. La première section définit, de manière concise, l'optimisation combinatoire et ses enjeux. Ceci permet de positionner les problèmes d'ordonnancement de la production présentés dans la seconde section. Cette dernière, décrit le problème d'ordonnancement, à savoir les contraintes, les types de ressources, les divers objectifs et la classification des problèmes d'ordonnancement ainsi que les outils de modélisation et de représentation associés. Finalement, et dans la troisième section, quelques définitions de base de la théorie des graphes sont rapportées.

- Le deuxième chapitre décrit dans un premier temps les problèmes d'ordonnancement auxquels nous nous intéressons dans les travaux de cette thèse, en présentant les différentes contraintes, les hypothèses du travail ainsi que les objectifs considérés. Dans un second temps la réduction de ce problème à un problème de multicoloration de graphes est présenté. Finalement ce chapitre rapporte les travaux de littérature dédiés à la résolution des problèmes d'ordonnancement par la multicoloration de graphes.
- Le troisième chapitre est dédié à la résolution de la version mono-objectif obtenue par l'agrégation des objectifs du problème de multicoloration de graphes considéré dans le chapitre précèdent. Pour ce faire une approche recuit simulé **ISA** est proposée avec un mécanisme de refroidissement amélioré afin de mieux guider la température en favorisant les paliers les plus prometteurs de l'espace de solution réalisables du problème. En outre, une technique de coloration des sommets basée sur la minimisation du gain perdu est proposée pour renforcer l'efficacité de l'approche proposée. Finalement, et afin de mesurer les performances l'approche **ISA**, des expérimentations sur instances de référence ont été réalisées. La méthode a montré une performance élevée par rapport à celles des autres méthodes de l'état de l'art sur de nombreuses instances.

- Dans le quatrième et dernier chapitre du manuscrit, nous considérons un problème de multicoloration de graphes modélisant un problème d'ordonnancement de tâches similaire au problème écrit dans le chapitre 2, où nous nous intéressons à optimiser deux objectifs distincts, à savoir : (i) maximisation du gain total des tâches exécutées (des sommets entièrement colorés), (ii) minimisation la somme de plus grande couleurs affectés à chaque sommet (des dates d'achèvement des tâches). Pour ce faire, la première section de ce chapitre rapporte les principales notions de base de l'optimisation multi-objectif, les méthodes de résolution des problèmes d'optimisation multi-objectif. Dans la seconde section une approche recherche tabou bi-objectif **BITS** est proposée. Finalement, dans la dernière section une comparaison de l'approche proposée avec l'une des approches de résolution des problèmes d'optimisation multi-objectif les plus performantes, à savoir, **NSGA-II** est présentée.

Chapitre 1

Généralités et notations

Sommaire

1.1 Introduction

Ce chapitre constitue une introduction générale aux problèmes d'ordonnancement liés à la gestion de production. Il présente diverses notions essentielles pour la compréhension de l'ensemble des travaux présentés dans cette thèse. La première section définie, de manière concise, l'optimisation combinatoire et ses enjeux. Ceci permet de positionner les différents problèmes d'ordonnancement, notamment le problème étudié. La deuxième section présente quelques notions qui interviennent dans la définition des problèmes d'ordonnancement, à savoir les classes et les typologies des problèmes d'ordonnancement ainsi que les outils de modélisation et de représentation associés. Finalement, dans la troisième section, quelques définitions de base de la théorie des graphes sont rapportées.

1.2 Notions fondamentales sur l'optimisation

1.2.1 L'optimisation combinatoire

L'optimisation combinatoire est une discipline qui occupe une place prépondérante dans la recherche opérationnelle et dans les mathématiques discrètes. Ceci est dû, non seulement, à son aptitude à modéliser une multitude applications pratiques, mais aussi, à sa capacité d'aborder les problèmes combinatoire les plus complexe à résoudre. L'objectif de ces problèmes combinatoires consiste à trouver une meilleure solution dans un espace fini et discret de solutions réalisables, qui respectent un ensemble de conditions, dites aussi contraintes. L'évaluation d'une solution est effectuée à l'aide d'une fonction dite fonction objectif. Une meilleure alternative (solution optimale) est une solution réalisable qui minimise ou maximise, selon le contexte, la fonction objectif (Maqrot, 2020).
Dans le reste de ce chapitre, nous nous contenterons de présenter les problèmes de minimisation ; les problèmes de maximisation obéissent aux mêmes règles, à quelques changements près.

Plus formellement un problème d'optimisation combinatoire peut être définit comme suit :

Definition 1.2.1. Sakarovitch (1984)
Un problème d'optimisation combinatoire est défini à partir d'un ensemble fini S et d'une application $f : S \rightarrow R$. Il s'agit de déterminer $\hat{s} \in S$ tel que :

$$f(\hat{s}) = Min_{s \in S}[f(s)].$$

A chaque problème d'optimisation on peut associer un problème de décision qui consiste à répondre à la question d'existence d'une solution pour laquelle la fonction objectif soit inférieure (resp. supérieure) ou égale à une valeur donnée, par un "OUI" ou par un "NON".

1.2.2 La théorie de la complexité

Un problème d'optimisation peut être résolu par plusieurs algorithmes, c'est pourquoi il est très important de pouvoir comparer l'efficacité des différents algorithmes de résolution de ce problème. Pour ce faire deux principaux critères sont généralement considérés : le premier est le temps de calcul, et le second est l'espace mémoire nécessaire à l'exécution de l'algorithme. Il est vrai que l'espace mémoire est un critère important pour l'évaluation de l'efficacité d'un algorithme, néanmoins, en pratique le temps de calcul est bien le paramètre principal mesurant l'efficacité d'un algorithme et c'est par rapport au temps de calcul que les problèmes d'optimisation combinatoire sont réputés si difficiles.

En effet, c'est la théorie de la complexité des algorithmes qui permet à l'optimisation combinatoire d'asseoir sur une base théorique solide, en donnant un sens précis au terme d'***algorithme efficace***, par l'analyse des coûts de résolution, notamment en terme de temps de calcul. Sur cette base les problèmes d'optimisation sont classés en plusieurs classes de complexité selon les algorithmes de résolution de leurs problèmes de décisions :

La classe P : Contient l'ensemble des problèmes d'optimisation pour lesquels il existe un algorithme déterministe qui s'exécute au pire des cas en

temps polynomial pour les résoudre. Ce temps d'exécution est de la forme $O(n^k)$ où n est un entier qui représente la taille du problème et k est une constante qui dépend de l'algorithme considéré. Généralement, le défi est de trouver un algorithme dont la constante k est la plus petite possible, i.e. le moins complexe possible.

La classe NP (non deterministic polynomial) : Contient l'ensemble des problèmes de décision pour lesquels il existe un algorithme non déterministe qui permet de le résoudre en temps polynomial ; ou encore pour lesquels il existe un algorithme polynomial qui permet de vérifier une solution (certificat) au problème (Liedlof, 2007).

Partant de l'idée de l'inclusion de la classe P dans la classe NP étant évidente, la raison principale qui laisse penser que $P \neq NP$ est l'existence d'un sous-ensemble qui contient les problèmes les plus difficiles : on les appelle le problèmes **NP-complets**. Une propriété importante de cette famille de problèmes est que si l'on savait résoudre l'un d'eux en temps polynomial, alors on pourrait tous les résoudre en temps polynomial, et par conséquent on obtiendrait la relation P = NP. Jusqu'à présent, de nombreux problèmes ont été montrés NP-complets et pour aucun d'entre eux un algorithme polynomial n'a été trouvé.

La théorie de complexité cherche à étudier les relations entre les problèmes dans le sens de leur difficulté dans le but d'obtenir une notion de hiérarchie de difficulté relative à leur résolution. Dans ce contexte, on fait appel à la réduction polynomiale pour montrer qu'un problème de décision est **NP-complet** (Karp, 1972), considérons deux problèmes π_1 et π_2, on dit que π_1 se réduit polynomialement à π_2, noté $\pi_1 \leq_p \pi_2$, s'il existe une fonction f polynomiale qui assure la transformation entre une instance de π_1 et une instance de π_2 telle que toute instance i de π_1 admet une solution si et seulement si l'instance $f(i)$ admet une solution pour le problème π_2 (Liedlof, 2007).

De manière pratique, pour démontrer la NP-complétude d'un problème, il n'est pas nécessaire de démontrer explicitement que pour tout $\pi' \in NP$, $\pi' \leq_p \pi$. En effet, grâce aux réductions polynomiales et par la transitivité des ré-

ductions. Parmi Les problèmes NP-complet nous citons le problème de la satisfiabilité d'un ensemble de clauses (SAT), c'est le problème connu comme NP-complet de référence Cook (1971). D'autres problèmes ont été montrés NP-complets, notamment, ceux cités dans les travaux de Garey (1979).

En résumé, la complexité d'un problème d'optimisation est liée à celle du problème de décision qui lui est associé. En particulier, si le problème de décision est NP-complet, alors le problème d'optimisation est dit NP-difficile.

1.2.3 Résolution des problèmes d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire revient à trouver une solution optimale dans un ensemble discret et fini. En théorie, il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure. Cependant, en pratique, l'énumération de toutes les solutions peut occasionner une explosion combinatoire, ce qui conduit à une croissance exponentielle du temps de recherche et c'est à cause de lui que les problèmes d'optimisation combinatoire sont réputés si difficiles. Cependant, les méthodes de résolution de problèmes appartiennent à deux grandes familles : méthodes exactes et méthodes approchées. Nous présentons dans ce qui suit les principales méthodes exactes et approchées les plus connues de la littérature.

1.2.3.1 Méthodes exactes

Le principe d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des combinaisons de l'espace de recherche. Parmi les méthodes exactes, nous trouvons les méthodes de séparation et évaluation (branch and bound) et la programmation dynamique.

Séparation et évaluation (Branch and bound)

Cette méthode a été introduite pour la première fois par Dantzig et al. (1954), et ce pour la résolution du problème du voyageur de commerce. Depuis, des centaines, voire des milliers de procédures de séparation et évaluation ont été

proposées pour divers problèmes d'optimisation combinatoire. Cette technique effectue un parcours de recherche arborescente pour une énumération implicite de toutes les solutions réalisables du problème de manière intelligente mais, l'analyse des propriétés du problème permet d'éviter l'énumération des solutions mauvaises.

La séparation consiste à décomposer le problème initial en sous-problèmes qui ont à leur tour la possibilité d'être décomposés. La structure de données de cette méthode est un arbre dans lequel chaque noeud représente un sous problème. L'évaluation d'un noeud de l'arbre de recherche a pour but de déterminer l'optimum de l'ensemble des solutions réalisables associé au noeud en question. Sinon, de prouver mathématiquement que cet ensemble ne contient pas de solution intéressante pour la résolution du problème et dans ce cas le noeud associé sera donc stérilisé.

L'efficacité de cette méthode dépend de trois importants paramètres :

- **la stratégie de recherche :** l'ordre dans lequel les noeuds de l'arbre sont explorés,
- **la stratégie de séparation :** comment l'espace de solution est partitionné pour produire de nouveaux sous-problèmes dans l'arbre,
- **les règles d'élagage :** quelles sont les règles à utiliser pour empêcher l'exploration des régions contenant des solutions non optimales.

Programmation Dynamique

C'est une méthode basée sur le principe de Bellmann (Bellman, 1957), initialement inventée par dans le but de résoudre les problèmes de chemins optimaux. Il s'agit d'une méthode d'énumération implicite très utilisée en optimisation combinatoire pour la recherche de solutions optimales dans un ensemble fini de solutions mais très grand. C'est une méthode ascendante qui commence par les sous-problèmes les plus petits et qui remonte vers les sous-problèmes de plus en plus difficiles. Cette technique construit une solution optimale du problème en combinant les solutions optimales de ses sous-problèmes en appliquant le principe d'optimalité de Bellman : "Dans

une séquence optimale, chaque sous-séquence doit aussi être optimale". Un exemple de ce type de problème est le plus court chemin entre deux sommets d'un graphe. L'idée de base est d'éviter de calculer deux fois la même chose, généralement en utilisant une table de résultats déjà calculés, remplie au fur et à mesure qu'on résout les sous problèmes.

1.2.3.2 Méthodes approchées

Les méthodes approchées représentent une alternative à la résolution de problèmes d'optimisation de grande taille lorsque les méthodes exactes n'arrivent pas à les résoudre en un temps raisonable. Le but est de trouver rapidement une solution réalisable tenant compte des contraintes du problème sans pour autant être en mesure de garantir l'optimalité.

Généralement, ces méthodes sont conçues pour un problème particulier, en s'appuyant sur sa structure propre, dans ce cas on parle de méthodes heuristiques. En outre, il existe des approches fondées sur des principes généraux, notées les métaheuristiques, ces approches partent d'une ou plusieurs solutions initiales qu'elles cherchent à améliorer à chaque étape.

Les méthodes heuristiques (constructives)

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. Leur principe général est d'intégrer des stratégies de décision pour construire une solution proche de l'optimum. Les heuristiques sont souvent utilisées pour des problèmes particuliers, de sort qu'une méthode qui est destinée à résoudre un problème donné ne peut être utilisée pour résoudre un autre problème.

Parmi les heuristiques les plus connues dans la littérature les algorithmes gloutons (greedy algorithms), sont des algorithmes pour lesquels, à chaque itération, on fixe la valeur d'une (ou plusieurs) des variables décrivant le problème sans remettre en cause les choix antèrieurs. Le principe est donc de partir d'une solution incomplète (éventuellement totalement indéterminée)

que l'on complète en effectuant des choix définitifs, à chaque étape, on traitant une partie des variables sur lesquelles on ne revient plus.

Les méthodes métaheuristiques (amélioratrices)

Une métaheuristique est une stratégie générale, applicable à un grand nombre de problèmes. L'idée de ces approches est de structurer l'ensemble des solutions en terme de voisinage, et d'explorer cet ensemble en partant d'une ou plusieurs solutions initiales et en choisissant à chaque itération une ou plusieurs solutions voisines d'une ou plusieurs solutions courantes. Le voisinage d'une solution s est l'ensemble des solutions qui peuvent être atteintes à partir de s en un seul "mouvement", un mouvement étant par exemple, le changement de valeur d'une variable. Dans ce qui suit, nous décrivons les métaheuristiques qui seront utilisées dans la suite pour la résolution des problèmes étudiés dans ce manuscrit.

- **Recherche Tabou :**

  Cette métaheuristique d'optimisation combinatoire a été développée par Glover (1986), le principe est d'effectuer des mouvements d'une solution s à une meilleure solution s' appartenant au voisinage de s. Ce voisinage peut être examiné complètement ou partiellement. Il arrive à cette méthode de ne pouvoir trouver une amélioration dans le voisinage exploré, dans ce cas la recherche tabou accepte des solutions voisines dont la qualité est moins bonne que la solution courante, où elle se dirige alors vers la solution voisine qui dégrade le moins possible la fonction objectif. Cette dégradation que l'on espère être temporaire, devrait permettre à l'algorithme d'atteindre des régions prometteuses.

  Cependant, l'algorithme tabou risque de boucler en revisitant des solutions antérieures pendant l'itération suivante. Pour éviter ce phénomène, l'algorithme utilise une structure de mémoire dans laquelle il stocke les mouvements interdits à chaque étape afin de ne pas obtenir une solution déjà rencontrée récemment. Ces mouvements qui sont interdits à une itération donnée sont dits tabous, d'où le nom attribué à cette méthode.

Le caractère tabou d'un mouvement doit être temporaire afin de donner une plus grande flexibilité à l'algorithme en lui permettant de remettre en question les choix effectués, une fois que les risques de bouclage ont été écartés.

La façon la plus simple de mettre en oeuvre ce système d'interdiction consiste à garder en mémoire les derniers mouvements effectués et à empêcher l'algorithme de faire les mouvements inverses à ces derniers mouvements. Ces mouvements sont mémorisés dans une liste T, appelée liste taboue. La gestion de cette liste se fait de manière circulaire en remplaçant à chaque itération le mouvement le plus ancien de la liste par le mouvement courant.

Dans la recherche tabou, le compromis entre les mécanismes d'intensification et de diversification peut être géré par la taille de la liste tabou. Pour favoriser l'intensification de la recherche il suffit de diminuer la taille de la liste tabou, et contrairement, pour favoriser la diversification il s'agit d'augmenter la taille de cette liste.

- **Recuit Simulé :**
  Cette méthode d'optimisation a été proposée par Kirkpatrick et al. (1983), à partir de la méthode Metropolis (Metropolis et al., 1953) développée à l'origine pour modéliser les processus physiques. Cette méthode repose sur une analogie avec la thermodynamique. En effet, la recherche par un système physique des états d'énergie les plus bas est l'analogue formel d'un processus d'optimisation combinatoire.

  Le recuit est un processus physique de chauffage, où un système physique porté à une température assez élevée devient liquide dans ce cas le degré de liberté des atomes qui le composent augmente. Inversement lorsque l'on baisse la température le degré de liberté diminue jusqu'à obtenir un solide. Suivant la façon dont on diminue la température on obtient des configurations d'énergie différentes, avec une baisse brutale de la température, la configuration atteinte est le plus souvent un état

métastable, dont l'énergie est supérieure à celle du minimum absolu. Dans ce cas le système est en quelque sorte piégé dans ce minimum local. Dans le cas où la baisse de température est progressive le système évite de se piéger dans des vallées d'énergie élevée.

De manière identique à ce processus physique, l'lgorithme recuit simulé permet de sortir d'un minimum local en acceptant avec une certaine probabilité une dégradation de la fonction. Ainsi, la probabilité P qu'un système physique passe d'un niveau d'énergie E_1 à un niveau E_2 est donnée par :

$$P = e^{-\frac{\Delta E}{k_b.T}}$$

k_b : est la constante de Bolzmann, T : est la température du système.

Comme le montre cette formule la probabilité d'observer une augmentation de l'énergie est d'autant plus grande lorsque la température est élevée, donc au niveau du recuit simulé :

- Une diminution de la fonction sera toujours acceptée ;
- Une augmentation de la fonction sera acceptée avec une probabilité définie selon une formule du type précédent.

Dans cette méthode, les mécanismes d'intensification et de diversification sont contrôlés par la température. En effet, la température décroît au fil du temps de sorte que la recherche tend à s'intensifier vers la fin de l'algorithme.

1.3 Problèmes d'ordonnancement

1.3.1 Definitions et notations

Les problèmes d'ordonnancement sont des problèmes d'optimisation combinatoire dans lesquels certaines tâches doivent être réalisées en utilisant les

ressources dont elles ont besoin, et ce tout en respectant les contraintes exigées. En d'autres termes un tel problème consiste à programmer dans le temps une allocation faisable des ressources aux activités. La qualité d'un ordonnancement est mesurée par divers critères d'optimisation qui sont en général des fonctions de délais d'exécution des activités et des quantités de ressources utilisées.

Les définitions de l'ordonnancement sont nombreuses dans la littérature. Nous en citons celles proposées dans Pinedo (2008), où il a présenté une synthèse sur les différents travaux relatifs à l'ordonnancement. Dans cet ouvrage une définition à cette famille de problèmes est donnée comme suit : *"L'ordonnancement est un processus décisionnel qui est utilisé régulièrement dans de nombreuses industries de production de biens et de services. Il traite l'allocation des ressources aux tâches sur des périodes de temps à determiner, et son but consiste à optimiser un ou plusieurs objectifs".*

Les différentes formes que les éléments des problèmes d'ordonnancement peuvent prendre ont permis à cette famille de problèmes de toucher à plusieurs domaines : l'informatique (les processeurs, ...), la construction (suivi de projet...), l'industrie (gestion de production, problèmes de logistique,...), l'administration (emplois du temps....). D'une manière générale les éléments d'un problème d'ordonnancement, à savoir les tâches, les ressources, les contraintes et les objectifs à optimiser peuvent être décrits comme suit :

Les tâches : Une tâche est une entité élémentaire de travail (job) localisée dans le temps par une date de début t_i (start time) et une date de fin c_i (completion time), dont la réalisation est caractérisée par une durée positive p_i (processing time) telle que $p_i = c_i - t_i$. D'autres caractéristiques relatives à l'exécution d'une tâche sont définies ainsi :

- Date de disponibilité r_i (release date) qui correspond à la date de début au plus tôt.
- Date d'échéance d_i (due date) qui correspond à la date de fin au plus tard.

- Poids w_i (weight) qui représente le facteur d'importance de la tâche i.

Les ressources :

Une ressource est un moyen technique ou humain utilisé pour la réalisation des tâches. Plusieurs types de ressources sont distingués :

- **Les ressources renouvelables** : Une ressource est dite renouvelable si après avoir été utilisée par une ou plusieurs opérations, elle est à nouveau disponible en même quantité (machines, personnels, équipements, etc).
- **Les ressources non-renouvelables** (ou consommables) : Une ressource est non renouvelables si sa disponibilité décroît après avoir été allouée à une opération (la matière première, budget, etc).
- **Les ressources disjonctives** (ou non-partageables) : il s'agit des ressources qui ne peuvent exécuter qu'une seule opération à la fois (machine, robot manipulateur, etc).
- **Les ressources cumulatives** (ou partageables) : il s'agit des ressources qui peuvent être utilisées par plusieurs opérations simultanément (ouvriers, poste de travail, etc.).

Les contraintes :

les contraintes représentent les conditions à respecter lors de la réalisation des tâches, deux types sont distingués, à savoir, les contraintes temporelles et les contraintes ressources :

- **Les contraintes temporelles** : décrivent les interdépendances temporelles entre les opérations, elles contiennent :
  - Les contraintes de dates limites : c'est des contraintes imposées individuellement à chaque opération. Par exemple, l'opération i ne peut débuter avant une date (livraison de matière première, etc.).
  - Les contraintes de précédence : c'est des contraintes qui relient la date de début ou la date de fin de deux opérations par une relation linéaire, c'est des contraintes qui décrivent le positionnement relatif de certaines opérations par rapport à d'autres.

- **Les contraintes de ressources** : traduisent l'utilisation et la disponibilité des ressources utilisées par les opérations, deux types sont distinguées, à savoir, contraintes cumulatives et contraintes disjonctives :
  - Les contraintes disjonctives : ces contraintes imposent la non-réalisation simultanée de deux opérations sur la même ressource.
  - Les contraintes cumulatives : ces contraintes expriment le fait qu'à tout instant, le total des ressources utilisées ne dépasse pas une certaine limite fixée.

Les objectifs :

Ce sont les critères d'évaluation ou bien les indicateurs de performance sur lesquels se base le choix d'un ordonnancement satisfaisant. En ordonnancement, les critères à optimiser consistent à minimiser ou maximiser une fonction objectif. Cette fonction objectif est généralement liée aux temps, aux ressources ou bien aux coûts.

- **les objectifs liés au temps** : c'est la catégorie des objectifs les plus étudiés en optimisation, parmi les plus classiques, nous pouvons citer :
  - Le temps total d'exécution (makespan) : défini par $C_{max} = max_{i \in E} C_i$ (E désigne l'ensemble d'opération) qui représente la date de fin du job le plus tardif.
  - La somme des dates d'achèvement des jobs (total flow time ou total completion time) : définie par $\sum_{i \in E} C_i$.
  - Le retard algébrique maximal (lateness) : défini par $L_{max} = max_{i \in E} L_i$, tel que $L_i = C_i - d_i$ est le retard algébrique pour chaque opération.
  - Le nombre pondéré de jobs en retard $N_T = \sum_{i=1}^{i=n} U_i$.
  - La somme pondérée des dates de fin d'exécution $\sum_{i=1}^{i=n} w_i \, C_i$.
  - La somme pondérée des retards absolus $\sum_{i=1}^{i=n} w_i \, T_i$.
- **Les objectifs liés aux ressources** : nous citons comme exemple :
  - Maximisation de la charge d'une ressource e.g., maximiser l'utilisation de la machine ayant une moindre consommation d'énergie.

  - Minimisation de nombre de ressources nécessaires pour réaliser un ensemble d'opérations.

- **Les objectifs liés au coût :** ces objectifs consistent généralement à minimiser les coûts de lancement, de production, de stockage, ou de transport.

En général, l'objectif souhaité dans un problème d'ordonnancement quelconque définit le critère à optimiser. Lorsque cet objectif renvoie un seul critère, il s'agit bien du cas d'optimisation mono-objectif, le cas le plus fréquent dans la littérature des problèmes d'ordonnancement. Cependant, lorsque l'objectif renvoie simultanément plusieurs critères à la fois, nous parlons dans ce cas d'optimisation multi-objectif. Cette dernière est devenue plus en plus important face à l'évolution et à la concurrence des systèmes de production.

1.3.2 L'ordonnancement dans un atelier

Dans un atelier de production les machines de producion représentent les ressources, il s'agit bien de ressources renouvelables. Dans la plupart des cas, ces machines ne réalisent qu'une seule opération à la fois, on parle alors de ressources disjonctives. Les problèmes d'ordonnancement dans un atelier peuvent être classés selon la configuration des machines ou bien selon l'ordre d'utilisation des machines pour la fabrication d'un produit donné.
La première catégorie classe ces problèmes, en fonction de la configuration des machines considérées :

- *Machine unique :* tous les jobs sont à exécuter sur la même machine.
- *Machines dédiées :* plusieurs machines, chacune étant spécialisée pour l'exécution de certaines opérations.
- *Machines parallèles :* plusieurs machines, tous les jobs peuvent être exécutés par cesmachines.

La deuxième catégorie classe ces problèmes, en fonction du passage des opérations sur les différentes machines, à savoir, Flow-shop, Job-shop et Open-shop (Figure 1.1) :

- Atelier à cheminement unique (Flow Shop) : Dans ce cas, tous les jobs utilisent les mêmes machines selon la même séquence. Autrement dit, l'atelier dispose de m machines en série, et tous les jobs doivent passer dans le même ordre sur les différentes machines (cas d'une chaîne de fabrication).
- Atelier à cheminement multiples (Job Shop) : Le problème du job shop se différencie du flow shop par un élément essentiel : le cheminement des jobs n'est pas unidirectionnel. Tout job est une succession ordonnée d'opérations, et chaque job ou famille de jobs possède une gamme opératoire spécifique.
- Atelier à cheminement libre (Open Shop) : Dans le cas d'un atelier de type open shop, les opérations de chaque job peuvent être exécutées dans n'importe quel ordre (les gammes des produits sont libres). Il n'y a donc pas de contrainte de précédence entre deux opérations d'un même job.

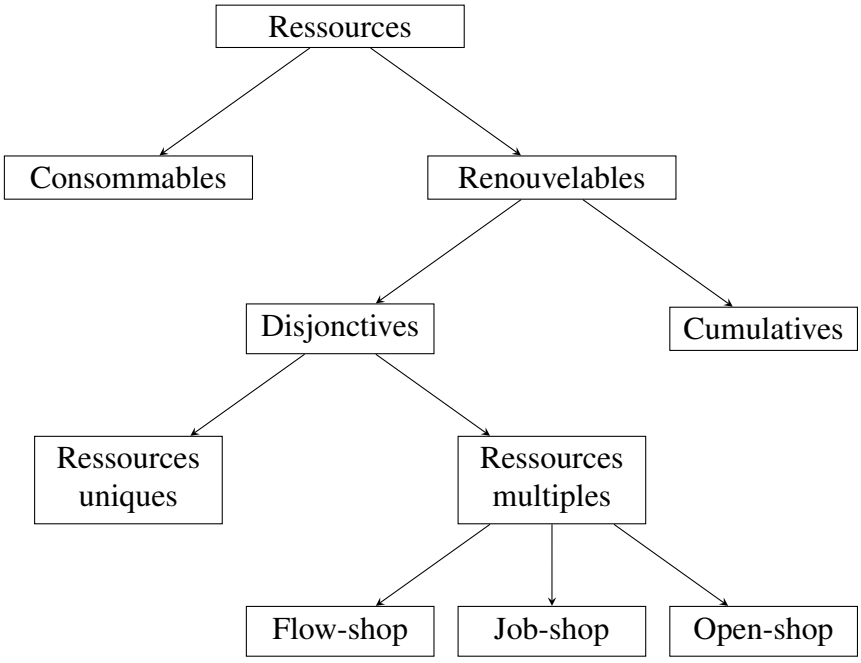

FIGURE 1.1 – Classification des ressources dans un atelier

1.3.3 Représentation des problèmes d'ordonnancement

La diversité des processus de production dans la pratique fait que les problèmes d'ordonnancement dans un atelier peuvent prendre différentes formes. Dans le but de les différencier Graham et al. (1979) ont proposé une méthode permettant une distinction de ces problèmes. Cette méthode se base sur trois paramètres notés $\alpha \mid \beta \mid \gamma$ signifiant l'environnement machines, les contraintes et les critères respectivement.

Le champ α désigne l'environnement machines. Il se compose en deux sous-champs α_1 α_2, le premier indique le type de problème étudié (problème à machines parallèles, Job Shop,...) tandis que le second précise le nombre de machines. Le champ β décrit les contraintes liées à l'exécution des tâches. Ce champ peut être vide $\emptyset$ là où aucune contrainte n'est imposée et peut contenir une concaténation de 1 à k sous-champs comme présentés dans le tableau 1.1. Finalement, le champ γ spécifie le critère à optimiser.

Champ		Sous champs	Notation
α	α_1	Type de machine	$\{\emptyset, 1, P, Q, R, F, J, O, FH, JF, OG\}$
	α_2	Nombre de machines	$\{\emptyset, m\}$
β	β_1	Mode d'exécution des jobs	$\{\emptyset, pmtn\}$
	β_2	Ressources supplémentaire	$\{\emptyset, res\}$
	β_3	Relation de précedence	$\{\emptyset, prec, tree, chains\}$
	β_4	Dates de disponibilité	$\{\emptyset, r_i\}$
	β_5	Durées opératoires	$\{\emptyset, p_i = p\}$
	β_6	Dates d'échéance	$\{\emptyset, d_i\}$
	β_7	Propriété d'attente	$\{\emptyset, n\ w\ t\}$
γ			$\{C_{max}, \sum w_i C_i, L_{max}, T_{max}, \sum w_i T_i, \sum U_i, \sum w_i U_i\}$

TABLEAU 1.1 – Classification des problèmes d'ordonnancement

On remarque sur le tableau 1.1 que les différentes notations utilisées pour les valeurs des champs α, β et γ sont généralement sous forme d'abréviations ayant chacune d'elles un sens particulier. Dans les tableaux 1.2, 1.3 et 1.4 on trouve une description de ces abréviations.

À titre d'exemple :

- $F_2 \| C_{max}$: problème d'ordonnancement d'un atelier de type Flow-Shop

Notation	Description
1	Problème à une seule machine
P	Problème à machines parallèles identiques
Q	Problème à machines parallèles uniformes
R	Problème à machines parallèles indépendantes
F	Flow-Shop
J	Job-Shop
O	Open-Shop
FH	Flow-Shop hybride
JF	Job-Shop flexible
OG	Open-Shop généralisé

TABLEAU 1.2 – Interprétation des notations du champ α_1

Notation	Description
$pmtn$	La préemption des opérations est autorisée
$prec$	Existence des contraintes de précedence entre les opérations
$tree$	Les contraintes de précédence sont données sous forme d'un arbre
$chains$	Les contraintes de précédence sont données sous forme de chaines
res	L'opération nécessite l'emploi d'une ou plusieurs ressources supplémentaires
nwt	Les opérations de chaque job doivent se succéder sans attente
$p_i = p$	Les temps d'exécution des tâches sont identiques et égaux à p
r_i	Une date de début au plus tôt est associée à chaque job i
d_i	Une date d'échéance est associée à chaque job i

TABLEAU 1.3 – Interprétation des principales notations possibles de sous-champs du champ β

à deux machines avec minimisation de makespan C_{max}.

- $P_m|pmtn|L_{max}$: problème à m machines parallèles identiques où la préemption est autorisée, l'objectif est la minimisation du retard maximum L_{max}.
- $1|prec,\ r_i|\sum w_i C_i$: problème à une machine dont les tâches présentent une contrainte de précedence et elles ne sont disponibles qu'à la date r_i, l'objectif est de minimiser la somme pondérée des dates de fin des tâches.

1.3.4 Complexité des problèmes d'ordonnancement

Dans beaucoup de cas pratiques les problèmes d'ordonnancement montrent certaines ressemblances dans leurs configurations, ainsi, un algorithme développé pour résoudre un problème d'ordonnancement particulier peut être ap-

Notation	Expression	Description
C_{max}	$max_{i\in\{1,...,n\}}C_i$	La durée totale de l'ordonnancement
L_{max}	$max_{i\in\{1,...,n\}}C_i - d_i$	Le plus grand retard algébrique
T_{max}	$max_{i\in\{1,...,n\}}\{max(C_i - d_i, 0)\}$	Le plus grand retard réel
$\sum[w_i]C_i$	$\sum_{i\in\{1,...,n\}}[w_i]C_i$	La somme pondéré des dates de fin des tâches
$\sum[w_i]T_i$	$\sum_{i\in\{1,...,n\}}[w_i]\{max(C_i - d_i, 0)\}$	La somme pondéré des retards
$\sum[w_i]U_i$	$\sum_{i\in\{1,...,n\}}[w_i]\|\{J_i/C_i > d_i\}\|$	Le nombre pondéré des tâches en retard

TABLEAU 1.4 – Interprétation des principales notations du champ γ

pliqué à un autre problème d'ordonnancement (Pinedo, 2008). Par exemple, $1 \mid\mid \sum C_{max}$ est un problème particulier de $1 \mid\mid \sum w_i C_{max}$ et un algorithme résolvant $1 \mid\mid \sum w_i C_{max}$ peut être appliqué pour la résolution de $1 \mid\mid \sum C_{max}$. En terme de complexité on dit que $1 \mid\mid \sum C_{max}$ est réduit à $1 \mid\mid \sum w_i C_{max}$. En se basant sur ce concept, une hiérarchie de complexité (aussi nommée l'arbre de réduction) de certains problèmes d'ordonnancement peut être établie en fonction de l'environnement machines, aussi en fonction des critères. La Figure 1.3 (resp. Figure 1.2) présente une hiérarchie de la complexité en fonction de l'environnement machines (resp. les critères). Cet arbre s'interprète de la manière suivante : pour un environnement machines et pour un critère donné, si un problème est NP-difficile, tous ses successeurs dans l'arbre le sont également. Cependant il existe aussi beaucoup de problèmes qui ne sont pas comparables les uns aux autres. A titre d'exemple : $P_m \mid\mid \sum w_i T_i$ n'est pas comparable à $J_m \mid\mid C_{max}$. Une présentation plus large et complète concernant la hiérarchie existant entre les différents problèmes d'ordonnancement peut être trouvée dans (Pinedo, 2008; Blacewicz et al., 1996).

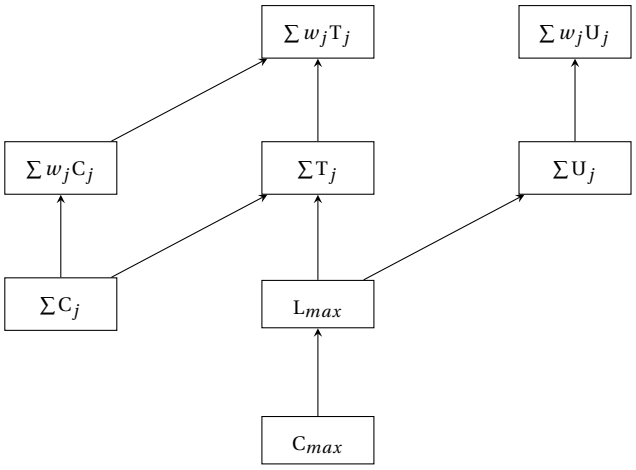

FIGURE 1.2 – Hiérarchie de complexité en fonction des critères

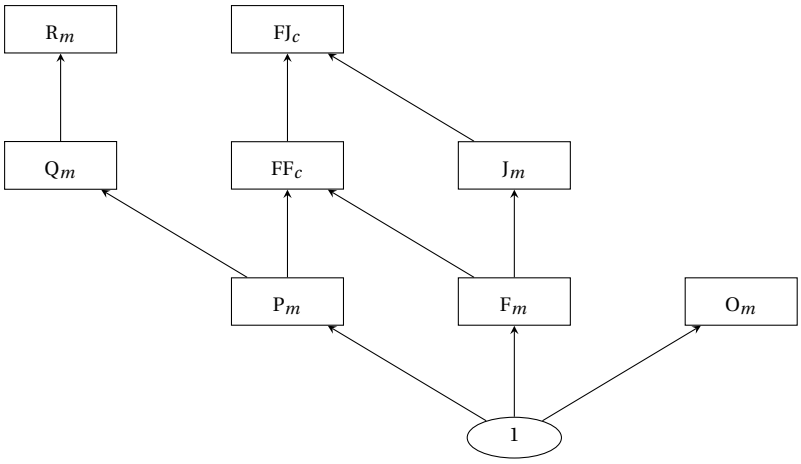

FIGURE 1.3 – Hiérarchie de complexité en fonction de l'environnement machines

1.4 Graphes et coloration

1.4.1 Définition et notations

La théorie des graphes constitue l'une des principales branches des mathématiques discrètes et de l'informatique fondamentale, son histoire a commencé par l'étude de certains problèmes, tels que celui des ponts de Königsberg, résolu par Euler au 1736, où les habitants de cette ville se demandaient s'il était possible, en partant d'un quartier quelconque de la ville, de traverser les sept ponts de la ville sans passer deux fois par le même et de revenir à leur point de départ.

Depuis le début du $20^{ème}$ siècle, la théorie des graphes s'est développée dans diverses disciplines en mathématiques, en informatique, et en sciences

sociales. De manière générale, la théorie des graphes peut être considérée comme un outil de modélisation permettant de représenter une structure d'un objet complexe en exprimant les relations entre ses éléments. Autrement, la théorie des graphes met au point un ensemble des techniques et outils mathématiques permettent de démontrer des propriétés, d'en déduire des méthodes de résolution et des algorithmes. En particulier, elle permet de représenter de nombreuses situations rencontrées dans des applications faisant intervenir des mathématiques discrètes et nécessitant une solution informatique par exemple en réseaux de télécoms, circuits électriques, réseaux de transport, et ordonnancement d'un ensemble de tâches.

En outre, la théorie des graphes peut être considérée comme un moyen de modélisation qui permet l'étude d'une grande variété de problèmes. Dans ce qui suit nous présentons les principaux aspects liés à cette branche.

Un graphe est une structure de données composée d'un ensemble de sommets, et d'un ensemble de relations entre ces sommets. Plus formellement, un graphe peut être défini comme suit :

Definition 1.4.1. Un graphe G est un couple (V, E) où V est l'ensemble non vide des points dont les éléments sont appelés sommets, et E est l'ensemble d'arêtes de G. Chaque arête uv est une paire de sommets u et v.

Par convention le nombre de sommets est noté n, et d'arêtes est noté m.
Un graphe orienté est un graphe G = (V, E), où chaque arrête entre couple de sommets u et v est dans une direction précise et est appelé arc, un arc uv est donc différent de l'arc vu.
Une boucle est une arête d'un graphe ayant pour extrémités le même sommet.
Un graphe est simple s'il ne comporte aucune boucle et que deux arêtes ne relient jamais la même paire de sommets.
Un graphe est planaire s'il a la particularité de pouvoir se représenter sur un plan sans qu'aucune arête (arc) n'en croise une autre.
Un graphe pondéré est un graphe G = (V, E), où les arêtes (ou les arcs) sont pondérées. Autrement dit, on associe une valeur positive ou négative à chaque

arête. Cela peut-être une distance : lorsque que l'on cherche un plus court chemin entre deux nœuds, il va de soit que la somme des pondérations des arêtes doit être aussi petit que possible.
Deux sommets u et v sont adjacents, si et seulement s'il existe une arête entre u et v dans E. Deux arêtes sont adjacentes si et seulement si elles ont une extrémité commune. Une arête d'un graphe ayant pour extrémités le même sommet est une boucle. Soit v un sommet de G, on appelle voisinage de v l'ensemble des sommets adjacents à v. Il est noté $\mathrm{N}(v)$.

Le degré d'un sommet v est le nombre de ses voisins. Il est noté $d(v) = |\mathrm{N}(v)|$. Les degrés minimum et maximum d'un graphe G sont notés $\delta(\mathrm{G})$ et $\Delta(\mathrm{G})$, respectivement. Si v est un sommet d'un graphe simple G d'ordre n, alors $0 \leq \delta(\mathrm{G}) \leq d(v) \leq \Delta(\mathrm{G}) \leq n-1$.

Une chaine de longueur $l+1$ est une succession de $l+1$ sommets $v_0,\ v_1,\ v_2,\ ...,\ v_l$ où $v_{i-1}\ v_i \in \mathrm{E}, \forall i = 1, ..., l$ et v_0 (resp. v_l) est l'extrémité initiale (resp. terminale) de la chaine.

Un cycle de longueur l est une séquence de sommets $v_0,\ v_1,\ ...,\ v_{l-1},\ v_0$ où $v_{i-1}v_i \in \mathrm{E}$ pour tout $i = 1,\ ...,\ l-1$ et $v_{l-1}\ v_0 \in \mathrm{E}$.

Un stable S est un sous ensemble de sommets deux à deux non adjacents, le nombre de stabilité $\alpha(\mathrm{G})$ d'un graphe G est la taille d'un stable maximum dans G.
Une clique C est un sous ensemble de sommets deux à deux adjacents, $\omega(\mathrm{G})$ (resp. $\theta(\mathrm{G})$) dénote la taille d'une clique maximum (resp. le nombre de partition minimum en cliques) du graphe G.

1.4.2 Représentation d'un graphe

Considérons un graphe $\mathrm{G} = (\mathrm{V},\ \mathrm{E})$, où n (resp. m) est le nombre de sommets (resp. d'arêtes) de G. Le codage usuel des graphes utilise soit une matrice d'adjacence, soit une liste d'adjacence pour chaque sommet et soit une matrice d'incidence.

Matrice d'adjacence

La représentation d'un graphe simple G par la matrice d'adjacence consiste en une matrice booléenne M de taille $n * n$, où pour deux sommets u et v, l'entrée M$[u][v]$ de la matrice détermine si l'arête (u, v) est présente ou pas dans le graphe. Cette représentation nécessite un espace mémoire de O(n^2). Dans le cas de graphes non orientés, la matrice est symétrique par rapport à sa diagonale. Donc, on peut seulement utiliser la composante triangulaire inférieure de la matrice d'adjacence.

Listes d'adjacence

La représentation d'un graphe G par les listes d'adjacence consiste en un tableau T de n listes, i.e., une liste pour chaque sommet, remplie par les sommets qui lui sont adjacents. Dans cette représentation, l'espace occupé par le graphe est proportionnel au nombre d'arêtes du graphe, et est de taille O$(n + m)$, où n est le nombre de sommets et m le nombre d'arêtes.

Matrice d'incidence

La représentation d'un graphe G par une matrice d'incidence qui consiste en une matrice booléenne M de taille $m * n$, où chaque ligne i correspond à un sommet i de G, et chaque colonne à une arête $u = (i,\ j)$ de G.

1.4.3 Coloration de graphes

Le problème de coloration de graphes est considéré comme l'un des champs les plus anciens de la théorie des graphes, il trouve son origine du problème des 4 couleurs posé par Francis Guthrie en 1852 comme suit : « *Est-il possible de colorier toute carte géographique avec au plus 4 couleurs de sorte que 2 régions ayant une frontière en commun aient des couleurs différentes* ? ». Un siècle s'est écoulé depuis l'énoncé de cette conjecture, en apparence très simple, plusieurs tentatives ont été faites pour la démontrer, ce qui a donné lieu à de nouveaux développements mathématiques ainsi qu'à une formulation du problème en termes de graphes. Ce n'est qu'en 1976 que deux chercheurs américains (Appel and Haken, 1977), de l'Université de l'Illinois, ont

pu répondre à cette conjecture, en affirmant que chaque graphe planaire sans boucles est 4-coloriable.

La coloration de graphes offre une immense capacité de modéliser des problèmes pratiques de diverses natures, depuis le placement de personnes autour d'une table ou de pièces sur un échiquier jusqu'aux différents problèmes d'ordonnancement et de planning de la vie de tous les jours (réservation de ressources, et logistique), et notamment dans le domaine du transport. Et ce, grâce à la coloration de plusieurs éléments d'un graphe : les sommets, les arêtes, les faces, un mélange de ces éléments, des sous-structures, et à ceci peut s'ajouter différentes contraintes, dont la plus courante est celle de la **propreté** : deux éléments voisins doivent avoir des couleurs différentes. Nous allons présenter les colorations "basiques" : la coloration de sommets, la coloration d'arêtes.

Etant donné un graphe G = (V, E), une **coloration propre de sommets** est une fonction qui attribue une couleur à chaque sommet de sorte que deux sommets adjacents aient des couleurs différentes. Le nombre minimum de couleurs d'une coloration propre des sommets de G est appelé le nombre chromatique de G et noté $\chi(G)$. Si $\chi(G) \leq k$, on dit que G est k-coloriable.
Une coloration est aussi une partition des sommets en ensembles indépendants (on parle de stables). Les couleurs, servant juste à définir quels éléments font partie d'un même stable, sont en général représentées par un numéro.

On peut aussi colorer les arêtes en évitant que des arêtes adjacentes aient la même couleur. Une « coloration des arêtes» d'un graphe G est une affectation de couleurs aux arêtes telles que les arêtes adjacentes soient de couleur différente. On cherche à déterminer une coloration utilisant le plus petit nombre de couleurs nécessaires pour colorer les arêtes d'un graphe G. Ce nombre est appelée « l'indice chromatique » de G et est noté $q(G)$.

On parle de la coloration impropre lorsqu'on relâche la contrainte consistant à colorer de manière différente les éléments adjacents d'un graphe. A titre d'exemple, certains problèmes d'allocation de fréquences sont mieux modélisés par le problème de coloration k-impropre (une coloration dans laquelle

un sommet a la même couleur qu'au plus k de ses voisins).

1.5 Conclusion

Ce chapitre a rappelé quelques notions fondamentales sur l'optimisation combinatoire, et quelques définitions de la théorie de complexité. Ensuite, il a présenté des généralités sur l'ordonnancement, en précisant ces principaux éléments, à savoir, les contraintes, les ressources, les différents objectifs et la classification des problèmes d'ordonnancement. A la fin, il rappelle quelques définitions de la théorie des graphes, en particulier la coloration des graphes, qui est considérée comme un outil de modélisation des problèmes d'optimisation combinatoire, notamment les problèmes d'ordonnancement.

Chapitre 2

Ordonnancement et multicoloration de graphes

Sommaire

2.1 Introduction

Dans ce chapitre, nous décrivons les différentes contraintes prises en compte dans le problème d'ordonnancement abordé dans cette thèse. Nous présentons ensuite un problème équivalent qui est le problème de multicoloration de graphes, puis une modélisation mathématique est donnée. Nous finalisons ce chapitre en présentons un état de l'art résumant les principaux travaux de la littérature en relation avec le problème étudié.

2.2 Problématique et notation

2.2.1 Énoncé du problème

Le problème d'ordonnancement de travaux (jobs) conflictuels que nous étudions dans cette thèse peut être énoncé comme suit : soit J un ensemble de n jobs dénotés J = $\{j_1,\ j_2,\ ...,\ j_n\}$ à éxécuter sur un ensemble M de m ressources dénotées M = $\{r_1,\ r_2, ...,\ r_m\}$. Chaque job peut être considéré comme une entité de travail élémentaire caractérisée par son temps de traitement p_i et son propre gain g_i obtenu lorsqu'il est entièrement traité, pour les jobs partiellement traités il n'y a pas de gain affecté.

La préemption lors du traitement des jobs est autorisée, ce qui signifie que l'exécution d'un job donné peut être arrêtée à tout moment et reprise plus tard. En effet, le temps de traitement de chaque job peut être divisé en plusieurs parties. Durant ces parties de temps, les jobs seront affectés à des machines éventuellement distinctes, tel sorte que les parties d'un même job doivent être assignées dans des périodes de temps qui ne se chevauchent pas. De plus, l'ensemble de ressources est divisé en deux types : **Les ressources cumulatives**, qui interviennent dans le traitement de tous les travaux et elles sont limitées par leur nombre ce qui limite le nombre de travaux qui peuvent être traités simultanément (i.e. les machines parallèles). **Les ressources disjonctives** représentent l'ensemble des ressources dédiées (i.e. certaines opé-

rations ne peuvent être traitées que par une ressource spécifique). Par conséquent, deux jobs nécessitant une même ressource disjonctive ne peuvent être traités en même temps.

Le traitement des jobs ordonnancés doit se terminer avant une échéance globale prédéfinie D. Cette échéance, correspond à la fin de l'horizon de l'ordonanncement. Il en résulte que seulement un sous-ensemble de jobs peut être réalisé sans retard. Le but est de définir un ordonnancement qui optimise trois objectifs. Le premier et le principal est de maximiser le gain total des jobs effectués, les deux autres objectifs considèrent la manière d'accomplir ces jobs, c'est-à-dire, minimiser le nombre d'interruptions et le temps total de réalisation des jobs.

2.2.2 Réduction du problème d'ordonnancement à la multicoloration de graphes

Le problème d'ordonnancement abordé peut être reduit à un problème de multicoloration de graphes défini par la donnée d'un graphe non oriénté G = (V, E), où V est l'ensemble des sommets et E est l'ensemble des arêtes. Chaque sommet v_i représente un job j_i, et il existe une arête (v_i, v_j) entre deux sommets v_i et v_j si les deux jobs correspondants sont incompatibles, c'est-à-dire nécessitant une même ressource critique. Le problème de la multicoloration des graphes consiste à attribuer un nombre prédéfini de couleurs à chaque sommet de sorte que deux sommets adjacents n'aient aucune couleur en commun.

Un problème de multicoloration de graphes peut être défini comme suit, étant donné un graphe G et une pondération des sommets $w : \mathrm{V(G)} \rightarrow \mathrm{N}$, une coloration du graphe pondéré (G, w) (où multicoloration de G) est une fonction $f : \mathrm{V(G)} \rightarrow \mathrm{P(N)}$ telle que chaque sommet v reçoit $w(v)$ couleurs et deux sommets adjacents ont des ensembles de couleurs disjoints.

La correspondance entre les objectifs du problème d'ordonnancement décrit dans la section section 2.2 et ceux du problème de multicoloration de graphes

est comme suit :

- Maximisation du gain total sur tous les sommets entièrement colorés.
- Minimisation du nombre d'interruptions dans la séquence de couleurs attribuées à chaque sommet.
- Minimisation de la somme de la gamme de couleurs attribuée à chaque sommet, sur tous les sommets entièrement colorés, où la gamme est définie comme la différence entre la plus grande couleur et la plus petite couleur.

Exemple d'illustration :

Considérons le problème d'ordonnancement Π composé de 7 jobs $j_1 - j_7$ à réaliser sur 2 ressources $r_1 - r_2$, chaque job j_i est caractérisé par un gain g_i et un temps de traitement p_i donnés comme suit dans la Tableau 2.1. Dans ce problème, tous les jobs planifiés doivent être terminés avant une échéance D = 10, et les jobs nécessitant la même ressource critique sont définis par l'ensemble : I = $\{(1,5);(2,7);(3,4);(3,5);(4,7);(5,6);(6,7)\}$

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7
g_i	6	4	3	7	2	1	4
p_i	5	2	4	4	3	5	2

TABLEAU 2.1 – Problème d'ordonnancement Π

Le graphe d'incompatibilité G = (V, E) modélisant les contraintes conflictuelles entre les jobs est donné sur la Figure 2.1, où l'ensemble des sommets V représente les jobs et l'ensemble des arêtes E représente les incompatibilités entre jobs.

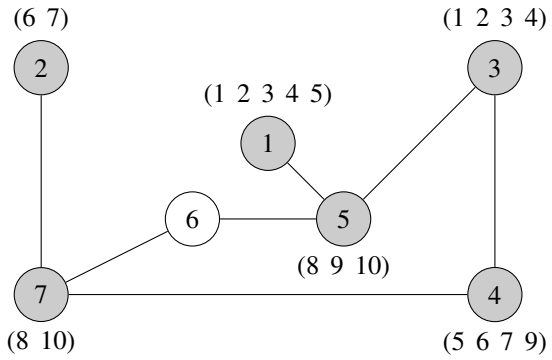

FIGURE 2.1 – Graphe d'incompatibilité G

A partir de la valeur de l'échéancé de planification D = 10, l'ensemble des couleurs disponibles est donné par C = $\{1,2,...10\}$. Selon le nombre de machines, chaque couleur peut être utilisée au maximum deux fois.

Enfin, une solution au problème Π est obtenue à partir de la coloration du graphe G (voir Figure 2.1), en attribuant p_i couleurs à chaque sommet v_i, de sorte que deux sommets adjacents ne puissent pas avoir les mêmes couleurs.

L'interprétation de la coloration du graphe en terme d'ordonnancement est détaillée dans la Tableau 2.2, où le job J_6 représenté par le sommet 6 n'est pas coloré car il ne reste plus de couleurs admissibles pour le colorer, i.e. job non planifié, et les 3 objectifs f_1, f_2 et f_3 sont égaux à 26, 2 et 22 respectivement

Unités de temps	1	2	3	4	5	6	7	8	9	10
Jobs processed on r_1	J_1	J_1	J_1	J_1	J_1	J_2	J_2	J_5	J_5	J_5
Jobs processed on r_2	J_3	J_3	J_3	J_3	J_4	J_4	J_4	J_7	J_4	J_7

TABLEAU 2.2 – Solution du problème d'ordonnancement Π

2.2.3 Modélisation mathématique

Formellement, le problème de multicoloration de graphe peut être modélisé par la programmation linéaire en nombres entiers. À cette fin, le modèle mathématique est donné comme suit (Thevenin et al., 2018a) :

Les indices :

i : indices des sommets $i = 1, ..., |V|$.

c : indice relatif aux couleurs $c = 1, ..., C$.

Les variables de décision :

$z_i = \begin{cases} 1 & \text{Si le sommet i est entièrement coloré,} \\ 0 & \text{sinon.} \end{cases}$

$X_{ic} = \begin{cases} 1 & \text{Si la séquence affectée au sommet i contient la couleur c,} \\ 0 & \text{sinon (notez que } X_{i0} = 0, \forall i \in V\text{).} \end{cases}$

$S_{ic} = \begin{cases} 1 & \text{Si la séquence de couleurs affectée au sommet i commence, ou est repris} \\ & \text{une interruption avec la couleur c,} \\ 0 & \text{sinon.} \end{cases}$

Max_i : est la plus grande couleur attribuée au sommet i (égale à 0 si'il n'est pas col

Min_i : est la plus petite couleur attribuée au sommet i (égale à 0 si'il n'est pas colo

La fonction objectif :

$$max(f1) = \sum_{i \in V} (z_i . g_i) \tag{2.1}$$

$$min(f2) = \sum_{i \in V} \sum_{c=1}^{D} S_{ic} \tag{2.2}$$

$$min(f3) = \sum_{i \in V} (Max_i - Min_i + z_i) \tag{2.3}$$

Les contraintes :

$$\sum_{c=1}^{\mathrm{D}} \mathrm{X}_{ic} = p_i.z_i \qquad i \in \mathrm{V} \qquad (2.4)$$

$$\sum_{i \in \mathrm{V}} \mathrm{X}_{ic} \leq l \qquad 1 \leq c \leq \mathrm{D} \qquad (2.5)$$

$$c.\mathrm{X}_{ic} \leq \mathrm{M}ax_i \qquad 1 \leq c \leq \mathrm{D}, \quad i \in \mathrm{V} \qquad (2.6)$$

$$c.\mathrm{X}_{ic} + \mathrm{D}.(1 - \mathrm{X}_{ic}) \geq \mathrm{M}in_i \qquad 1 \leq c \leq \mathrm{D}, \quad i \in \mathrm{V} \qquad (2.7)$$

$$\mathrm{D}.z_i \geq \mathrm{M}in_i \qquad i \in \mathrm{V} \qquad (2.8)$$

$$\mathrm{X}_{ic} + \mathrm{X}_{jc} \leq 1 \qquad 1 \leq c \leq \mathrm{D}, \quad (i,j) \in \mathrm{E} \qquad (2.9)$$

$$\mathrm{S}_{ic} \geq \mathrm{X}_{ic} - \mathrm{X}_{i(c-1)} \qquad 1 \leq c \leq \mathrm{D}, \quad i \in \mathrm{V} \qquad (2.10)$$

$$\mathrm{S}_{ic},\ z_i,\ \mathrm{X}_{ic} \in \{0,1\} \qquad 1 \leq c \leq \mathrm{D}, \quad i \in \mathrm{V} \qquad (2.11)$$

$$\mathrm{M}in_i,\ \mathrm{M}ax_i \in \mathbb{N} \qquad i \in \mathrm{V} \qquad (2.12)$$

Les trois termes de la fonction objectf correspondent respectivement à :

- Maximiser le gain total sur tous les sommets entièrement colorés (équation 2.1).
- Minimiser le nombre d'interruptions dans la séquence de couleurs attribuée à chaque sommet (équation 2.2).
- Minimiser la somme de la gamme de couleurs attribuée à chaque sommet, sur tous les sommets entièrement colorés (équation 2.3).

Les différentes contraintes du modèle mathématique sont interprétées comme suit :

- La contrainte (2.4) impose l'attribution de p_i couleurs à chaque sommet v_i entièrement coloré.
- La contrainte (2.5) impose que toute couleur c soit affecteé au maximum à l sommets.
- Les contraintes (2.6) - (2.8) définissent $\mathrm{M}in_i$ (resp. $\mathrm{M}ax_i$) la plus petite (resp. grande) couleur attribuée au sommet v_i.
- La contrainte (2.9) indique que deux sommets adjacents du graphe d'incompatibilité G ne peuvent pas partager une couleur commune.

- La contrainte (2.10) indique le début d'une séquence de couleurs affectée au sommet v_i ou un redémarrage en forçant S_{ic} à 1 lorsque $X_{ic} = 1$ et $X_{(i(c-1))} = 0$. Sinon, la soustraction est négative ou nulle et S_{ic} est forcée à la valeur 0 par la minimisation du second terme dans la fonction objectif.

2.3 Travaux en relation

Le problème d'ordonnancement de jobs conflictuels provient des applications dont la disponibilité des ressources est limitée. Dans un contexte pareil, certains jobs ont une demande spécifique pour des ressources données, comme des opérateurs qualifiés ou des outils dédiés. En effet, lorsque la demande totale de ces ressources dépasse l'offre, des conflits lors de l'exécution de certains jobs peuvent survenir, par conséquent, il ne serait pas possiblle à aucun moment de se chevaucher dans leur traitement. Ainsi, la coloration des graphes est l'une des branches de l'optimisation combinatoire les plus connus de pouvoir traiter ce type de problèmes et diverses autres applications pratiques du monde réel. Par exemple la modélisation des réseaux sociaux (Mosa et al., 2017), la parallélisation des calculs scientifiques qui s'exécutent dans des environnements distribués multi-GPU (Bogle et al., 2022; Zheng et al., 2020) , la modélisation des effets du bruit dans la synchronisation des oscillateurs à relaxation (Vaidya et al., 2021), l'affectation de fréquence (Orden et al., 2019; Mahapatra et al., 2019), l'élaboration des planning des examens (Abou Kasm et al., 2019), la prise des décisions stratégiques dans la conception des gares (Jovanović et al., 2020), et particulièrement dans les problèmes d'ordonnancement (Thevenin et al., 2018b; Kouider et al., 2017; Kouider and Ait Haddadène, 2021).

Plus précisément, dans la théorie de l'ordonnancement, plusieurs problèmes avec des jobs conflictuels peuvent être modélisés par la coloration de graphe, où les sommets du graphe représentent des jobs et une arête entre deux sommets représente une relation entre les jobs correspondants, ce qui signifie que

ces jobs ne peuvent pas être traités en même temps. Parmi les cas pratiques, Kheiri et al. (2021) proposent une méthode exacte utilisant le problème de coloration de graphes partitionnés pour construire un planning optimal des opérations chirurgicales, dont le but est de minimiser le nombre d'annulations en raison de l'indisponibilité des lits pour la récupération post-opératoire. D'autres exemples d'applications d'ordonnancement utilisant la coloration de graphes peuvent être trouvés dans (Gamache et al., 2007; Giaro et al., 2009).

Dans certains cas, les modèles de coloration de graphes présentent des limitations pour représenter diverses exigences imposées par des applications réelles, par exemple, il ne peuvent pas modéliser certains problèmes d'ordonnancement ayant à la fois des contraintes de précédence et des conditions d'incompatibilité. Pour ce type de problème d'ordonnancement, Sotskov and Tanaev (1976) ont introduit la coloration des graphes mixtes qui consiste à l'affectation d'entiers positifs (couleurs) aux sommets du graphe mixte tel que si deux sommets sont reliés par une arête alors leurs couleurs doivent être différents et si deux sommets sont reliés par un arc, les couleurs du sommet de départ ne doivent pas être supérieures aux couleurs du sommet de fin. Ce modèle général est capable de répondre à des contraintes de précédence supplémentaires. Parmi les travaux dans ce contexte, (Sotskov et al., 2001; Kouider et al., 2015, 2017) ont considéré la coloration de graphes mixtes pour le problème d'ordonnancement job-shop par unité de temps avec le critère de makespan, tandis qu'Al-Anzi et al. (2006) ont considéré le même problème avec le critère du temps de réalisation total.

De plus, dans la littérature, plusieurs problèmes du monde réel ont été réduits à des problèmes d'ordonnancement. Parmi ceux, Shukri et al. (2021) ont étudié la planification des tâches dans le cloud computing en proposant une version avancée de Multi-Verse Optimizer, qui améliore à la fois les algorithmes originaux de Multi-Verse Optimizer et Particle Swarm Optimization dans des environnements cloud. Bo et al. (2021) ont proposé un modèle de

planification collaborative à réponse rapide basé sur la surveillance en ligne du système de production cyber-physique pour résoudre le problème de la réparation à réponse rapide des perturbations dans la fabrication des essieux montés des trains à grande vitesse. Kessler et al. (2021) éont étudié le problème d'ordonnancement de la couronne (crown-scheduling), connu par l'approche d'ordonnancement statique pour des ensembles de jobs parallélisables avec une échéance commune, visant à minimiser la consommation d'énergie sur des processeurs parallèles avec une mise à l'échelle de la fréquence.

Le modèle de multicoloration de graphes est une extension de la coloration de graphes. Elle est proposée dans la littérature pour modéliser les problèmes d'ordonnancement de jobs avec des durées d'exécution différentes. Dans cette extension, chaque sommet se voit attribuer un ensemble de couleurs, où le nombre de couleurs attribuées à un sommet est spécifié par le temps de traitement de son job correspondant. Il est évident que dans un problème de multicoloration, les sommets adjacents ne peuvent pas recevoir des couleurs similaires. Par conséquent, les sous-ensembles de couleurs reçus par des sommets connectés doivent être disjoints. Le problème bien connu de la multicoloration est réservé au cas où il n'y a pas de restriction sur l'ensemble des couleurs que chaque sommet peut recevoir (hormis sa taille) et l'objectif est de minimiser le nombre de couleurs utilisées. Un problème différent est obtenu lorsqu'il est demandé à ce que les couleurs affectées à un sommet forment un intervalle contigu, il s'agit d'une affectation de couleurs non préemptive. Sinon, lorsque des couleurs affectées à un sommet forment un intervalle non contigu, il est fait référence à une affectation de couleurs préemptive. Différentes fonctions objectifs sont traitées dans divers problèmes de multicoloration, en particulier la minimisation de la somme des multicolorations, la somme de la plus grande couleur attribuée à chaque sommet, ou la plus grande couleur attribuée (en supposant que les couleurs correspondent à des nombres naturels). D'un point de vu complexité, le problème multicoloration NP-difficile de même que son problème réduit qui la coloration de graphes Karp (1972).

Un sommaire du problème de multicoloration et de ses applications dans les problèmes d'ordonnancement est donné dans Marx (2004). Parmi les travaux récents sur la multicoloration, Méndez-Díaz and Zabala (2010) proposent un programme linéaire mixte en nombres entiers (MILP) et une méthode branch-and-cut pour résoudre le problème de la multicoloration de graphes n'ayant qu'un nombre prédéfini de conflits. Ce problème concerne un environnement d'ordonnancement de la production dans lequel des ressources conflictuelles peuvent être partagées jusqu'à une certaine limite. Blöchliger and Zufferey (2013) proposent des méthodes exactes et des métaheuristiques pour résoudre le problème multicoloration qui modélise l'ordonnancement des jobs dans un environnement de machines parallèles avec préemption, pénalités d'incompatibilité et coûts d'affectation. Thevenin et al. (2017) ont étudié un problème d'ordonnancement de machines parallèles avec incompatibilité des jobs via la multicoloration du graphe, où le nombre de machines est limité et les tâches ont des temps de traitement différents. Chaque sommet du graphe nécessite plusieurs couleurs et le nombre de sommets de la même couleur est limité. Dans leur étude, ils ont considéré trois objectifs : le makespan, le nombre de préemptions et la somme des temps de traitement des jobs. Thevenin et al. (2018b) s'intéressent à une variante de multicoloration de graphes dans laquelle le nombre de couleurs disponibles est tel que tous les sommets ne peuvent pas être colorés. Dans leur problème, il existe une limite sur le nombre de sommets, auxquels on peut attribuer la même couleur. Un profit est associé à chaque sommet et deux objectifs sont considérés. La première consiste à maximiser le profit total sur tous les sommets colorés. Le deuxième objectif considère la séquence de couleurs attribuées à chaque sommet dans laquelle la plage et le nombre d'interruptions doivent être minimisés, où la plage correspond à la différence entre la plus grande et la plus petite des couleurs attribuées à un sommet.

2.4 Conclusion

Dans ce chapitre, nous avons présenté une description du problème d'ordonnancement de jobs conflictuels, où chaque job peut être considéré comme une entité de travail élémentaire caractérisée par son temps de traitement, son propre gain obtenu lorsqu'il est entièrement traité. La préemption lors du traitement des jobs est autorisée. Le traitement des jobs ordonnancés doit se terminer avant une échéance globale prédéfinie. Dans la seconde section une réduction du problème étudié à un problème de multicoloration de graphes et sa modélisation mathématique est présentée. La dernière section de ce chapitre rapporte les travaux de références liés au problème considéré.

Chapitre 3

Multicoloration de graphes pour l'ordonnancement mono-objectif : Approche de résolution

Sommaire

3.1 Introduction

Dans ce chapitre, nous nous intéressons à la résolution du problème décrit dans le chapitre précédent concernant la multicoloration de graphes. Ce problème d'optimisation qui modélise de nombreux application réelles, notamment le problème d'ordonnancement des jobs conflictuels qui nous préoccupe.

Dans la littérature diverses approches ont déjà été proposées afin de résoudre ce problème multiobjectif via des fonctions d'agrégations des objectifs. Ainsi, une approche basée sur le recuit est proposée, cette approche est dotée d'un mécanisme de refroidissement. Ainsi qu'une technique de coloration basée sur la minimisation du gain perdu lors de la coloration des sommets, dans le but de renforcer l'efficacité de l'algorithme proposé. Des expérimentations sur des instances de références sont réalisées dans le but de valider les performances de l'approche proposée.

3.2 Approche de résolution

Le recuit simulé (Simulated Annealing SA) est une technique stochastique pour approcher l'optimum global des problèmes d'optimisation combinatoire, c'est une approche issue de l'analogie avec la métallurgie, où le processus consiste à faire monter la température du solide à des valeurs élevées, puis à laisser refroidir lentement pour atteindre des états de basse énergie d'un solide.

L'approche d'optimisation des problèmes d'optimisation combinatoire recuit simulé est initialement introduite par Kirkpatrick et al. (1983), où l'idée de base utilise la procédure itérative de Metropolis et al. (1953), afin d'atteindre un état d'équilibre thermodynamique. C'est une procédure permet de sortir des minimas locaux avec une probabilité d'autant plus grande que la température est élevée. Quand l'algorithme atteint de très basses températures, les états les plus probables constituent en principe d'excellentes solutions au

problème d'optimisation (Algorithmes 3.1 & 3.2).

Algorithme 3.1 Algorithme du Metropolis

1: Evaluer la variation d'énergie associée à une transition élémentaire aléatoire de l'état courant i, d'énergie E_i, vers un nouvel état j, d'énergie E_j : $\Delta\mathrm{E}_{ij} = \mathrm{E}_i - \mathrm{E}_j$

2: Accepter la transition vers le nouvel état avec une probabilité P_{ij} où :

$$\begin{cases} \mathrm{P}_{ij}(\mathrm{T}) = 1 & si \quad \Delta\mathrm{E}_{ij} \leq 0 \\ \mathrm{P}_{ij}(\mathrm{T}) = e^{\Delta \mathrm{E/T}} & si \quad \Delta\mathrm{E}_{ij} > 0 \end{cases}$$

Pareillement, la méthode (SA) commence par une solution initiale générée aléatoirement ou construite avec une heuristique efficace. Ensuite, une solution voisine est générée en modifiant la solution courante avec une technique de perturbation. Après cela, une décision est prise de manière probabiliste pour remplacer ou non la solution courante par la solution voisine. Pour cela, la solution voisine est acceptée si sa qualité, évaluée par une fonction de coût, est améliorée. Sinon, la solution est acceptée avec une probabilité $\mathrm{P} = e^{\Delta \mathrm{E/T}}$, où ΔE est la variation de coût entre la solution courante et la solution voisine et T est la température. En effet, la température initiale et la fonction de décroissance de la température sont des paramètres déterminants de l'efficacité du recuit simulé. D'autres paramètres de contrôle ne sont aussi pas moins décisifs, notamment la procédure de construction d'une solution voisine, cette dernière devrait garantir une exploration maximale de l'espace de solutions réalisables du problème.

Partant de l'importance de ces paramètres sur l'efficacité de la méthode recuit simulé pour résoudre les problèmes d'optimisation combinatoire, nous avons proposé une approche recuit simulé améliorée (Improved Simulated Annealing **ISA**) (Baaziz et al., 2023) dotée d'un nouveau mécanisme de refroidissement modifié et d'une technique de construction de solution voisine, et ce pour résoudre le problème de multicoloration de graphe étudié dans le chapitre précédent.

Algorithme 3.2 Algorithme du recuit simulé

1: Définir la fonction objectif (f).
2: Choix des mécanismes de perturbation d'une configuration ΔS.
3: Tirer une configuration aléatoire S.
4: Calculer l'énergie associée à cette configuration E.
5: Initialiser la température T_0.
6: **Tant que** Conditions d'arrêts pas satisfaites **faire**
  Tant que l'équilibre thermodynamique pas atteint **faire**
   Tirer une nouvelle configuration S′.
   Si (f(S′) < (S))
    $f_{min} = f$(S′)
    S_{opt} = S′
   Sinon
    Appliquer la règle de Metropolis.
   fin Si
  fin tant que
  Décroître la température.
 fin tant que
7: Afficher la solution optimale.

3.2.1 Mécanisme de refroidissement

Dans l'algorithme recuit simulé, la température agit comme un paramètre de contrôle, où des perturbations des solutions sont apportées à chaque niveau de température pour permettre à l'énergie de se stabiliser à un équilibre thermique. Ce process est répété en diminuant la température jusqu'à ce que la température minimale soit atteinte. La difficulté la plus importante dans la mise en œuvre de l'algorithme est qu'il n'y a pas d'analogie évidente pour le choix de la température initiale et la perturbation de la température par rapport à un paramètre libre dans le problème combinatoire Ramesh et al. (2021).

Le réglage de la décroissance de la température ou bien le mécanisme de refroidissement est très important dans l'algorithme du recuit simulé. Une décroissance rapide de température entraine un risque de piéger l'algorithme dans un minimum local, alors qu'une lente décroissance entraîne une forte divergence de l'algorithme. D'un point de vue théorique, la convergence théo-

rique du recuit simulé donné par la loi logarithmique Hajek (1988) :

$$T_k = \frac{\mu}{log(1+k)}$$

Où k est le nombre de paliers de température effectués, μ une constante positive égale à la profondeur maximale des minimas locaux. En pratique, cette règle induit un temps de calcul très important, donc la décroissance géométrique est souvent utilisée :

$$T_{k+1} = \alpha.T_k$$

Où α est une constante dans l'intervalle $[0,1]$.

Dans notre travail (Baaziz et al., 2023) nous avons proposé un mécanisme de refroidissement favorisant l'acceptation des nouvelles solutions, cela renforce l'algorithme pour atteindre l'optimum global plutôt que de rester bloqué à une valeur optimale locale. La distinction est dans la variation de température, qui a un rôle important dans l'équation de probabilité d'accepter ou de rejeter une pire solution pour sortir des minimas locaux. Pour favoriser la sortie des minimas locaux, nous proposons une approche modifiée de recuit simulé qui prend en compte la qualité de la solution dans la variation de température. Dans notre méthode, à la place du mécanisme de refroidissement classique qui est monotone non croissant, nous utilisons un mécanisme de refroidissement qui maintient la température inchangée si le coût de la nouvelle solution n'est pas amélioré à chaque étape de température.

Le pseudo-code de l'Algorithme 3.3 montre l'aperçu de notre approche recuit simulé améliorée **ISA** proposée pour la multicoloration de graphes, qui modélise le problème d'ordonnancement.

3.2.2 Technique de coloration

Un second paramètre qui agit fortement sur l'efficacité du recuit simulé est bien la façon de construire de nouvelles solutions voisine S_v. Ceci dépend non seulement de la technique de codage des solutions qui facilite la mise

Algorithme 3.3 Recuit simulé pour la multicoloration de graphes **ISA**

Entrées
$G = (V, E)$: graphe d'incompatibilité correspondant au problème d'ordonnancement
Nb, T_0, L, α, K_B : Paramètres de l'algorithme
Sorties S^* : la meilleure solution du problème
Début
Fixer les paramètres de l'algorithme
Construire une solution initiale S_0
$S \leftarrow S_0$; $S^* \leftarrow S_0$
Tant que le nombre de paliers de température n'est pas atteint **faire**
$T \leftarrow \alpha.T$
$Cout \leftarrow F(S)$
Tant que la langueur du palier de température n'est pas atteinte **faire**
Générer une solution voisine S_v
$\Delta F \leftarrow (S_v) - F(S)$
Si $(\Delta F \leq 0)$
$S \leftarrow S_v$
Si $(F(S) < F(S^*))$
$S^* \leftarrow S$
fin Si
fin Si
Si $(\Delta F > 0)$
$Prob(\Delta F, T) \leftarrow exp^{(-\Delta F)/(K_B.T)}$
Générer q uniformément dans l'intervalle $[0, 1]$
Si $(q < Prob(\Delta F, T))$
$S \leftarrow S_v$
fin Si
fin Si
fin tant que
Si $(Cout < F(S))$
$T \leftarrow T/\alpha$
fin Si
fin tant que
Fin

en oeuvre de l'algorithme, mais aussi sur la façon de colorer un sommet. Ce processus devrait garantir à l'algorithme recuit simulé d'explorer au mieux l'espace des solutions réalisables. Dans ce contexte, plusieurs stratégies de coloration ont été proposées dans la littérature. Nous citons celles proposées dans les travaux de Thevenin et al. (2013, 2018b), où les auteurs ont adopté un algorithme de recherche tabou pour la multicoloration de graphes en utilisant plusieurs techniques de coloration que nous décrivons comme suit :

- Technique du plus grand ensemble contigu : commencez par attribuer le plus grand ensemble de couleurs continues parmi les couleurs admissibles ; continuez à attribuer les couleurs une par une en minimisant le nombre d'interruptions.
- Technique des plus petites couleurs : à chaque étape, la couleur la plus petite est attribuée.
- Technique R^{SD} : le but est d'attribuer une couleur à un sommet qui entraîne la plus petite augmentation du degré de saturation de ses sommets adjacents qui ne sont pas colorés.
- Technique R^{Enf} : cette technique est utilisée lorsque le nombre de couleurs n'est pas suffisant. D'abord, toutes les couleurs admissibles sont attribuées, puis les autres couleurs sont attribuées en décolorant à chaque étape un sommet voisin avec le plus petit gain.
- Technique R^{Exact} : procédure de coloration optimale pour les deux objectifs : maximiser le gain total sur tous les sommets entièrement colorés et minimiser la somme du nombre d'interruptions dans la séquence de couleurs attribuée à chaque sommet sur tous les sommets entièrement colorés.

L'étude comparative présentée dans leurs travaux a montré la supériorité de la technique R^{SD} sur les autres techniques de coloration. Dans notre démarche, et afin d'attribuer les couleurs disponibles aux différents sommets du graphe d'une manière efficace, tout en respectant les contraintes de conflits, nous adoptons une technique qui colore à chaque étape un sommet du graphe d'incompatibilité G, en commençant par lui attribuer les couleurs qui ne peuvent

pas être utilisées par ses voisins, c.-à-d. les couleurs qui sont déjà attribuées aux voisins de ses voisins, si le nombre de ces couleurs est inférieur à p_i alors on lui attribue les autres couleurs avec un ordre minimisant le gain perdu, c.-à-d. les couleurs qui peuvent être utilisées par les voisins ayant le plus faible gain g_i.

L'algorithme 3.4 décrit les différentes étapes de la technique de coloration proposée.

Algorithme 3.4 Stratégie de gain maximum pour la coloration des sommets

Entrées
G = (V, E) : graphe d'incompatibilité correspondant au problème d'ordonnancement
p : Vecteur de nombre de couleurs disponible
g : Vecteur des gains de chaque sommet
l : Nombre de répétition des couleurs
v_i : Sommet à colorer
Sorties : Les couleurs affectées au sommet v_i
Début
Énumérer toutes les couleurs possibles à affecter au sommet v_i.
si (le nombre de couleurs possibles est superieur à p_i)
- Calculer pour chaque couleur c_i parmi les couleurs possibles le gain perdu (i.e. la somme des gains des somets voisins à v_i qui ne sont pas encore coloré et peuvent être coloré par la couleur c_i)
- Colorer le sommet v_i avec p_i couleurs qui sont assignées à moins de l sommets et ayant un gain perdu minimum
sinon
- Le sommet v_i ne sera pas coloré
finsi
Fin

3.3 Expérimentations et résultats

3.3.1 Instances utilisées et mise en oeuvre de l'algorithme

Les instances utilisées dans les expérimentations sont tirées des travaux de Thevenin et al. (2018b). Dans toutes ces instances, le nombre de sommets n et la densité du graphe d (probabilités d'existence une arête entre deux sommets) sont donnés comme suit : au total, 75 instances ont été testées avec $n \in \{10, 25, 50, 100, 500\}$ et $d \in \{0.2, 0.5, 0.8\}$.

Pour chaque paire (n, d), cinq instances sont générées et étiquetées avec les lettres a, b, c, d et e. Le nombre de couleurs p_i du sommet i est un entier choisi aléatoirement dans l'intervalle $[1, 10]$. Le gain g_i est lié à p_i car $g_i = \beta.p_i$, où β est un nombre aléatoire choisi dans l'intervalle $[1, 20]$. Pour chaque instance, la valeur du paramètre l est fixée à 30 et le nombre de couleurs disponibles D est choisi suffisamment petit pour éviter de colorer tous les sommets. Les instances utilisées sont classées en deux types : les petites instances avec $n \in \{10, 25, 50\}$, et les grandes instances avec $n \in \{100, 500\}$.

L'approche proposée est implémentée en C++ et est exécutée sur un processeur Intel Quad-core i7 avec 8 Go de mémoire RAM DDR3. Le temps CPU maximum autorisé (en secondes) est de $60.n$ (n est un nombre de sommets), et 5 exécutions pour chaque instance sont effectuées.

3.3.2 Impact du mécanisme de refroidissement proposé

Dans le but d'évaluer l'impact du mécanisme de refroidissement proposé, nous avons comparé deux approches recuit simulé similaires à l'exception du mécanisme de refroidissement, où la première est dotée le mécanisme de refroidissement modifié **ISA**, et la deuxième contient le mécanisme de refroidissement classique **SA**, (c'est-à-dire : nous maintenons les autres paramètres des deux approches similaires). Afin de valider cette comparaison, nous avons choisi 15 instances de taille 100 sommets pour analyser et évaluer les performances.

Les résultats obtenus par les deux approches sont présentés en détail dans la Table 3.1, où la colonne **"Best"** contient les meilleures valeurs des trois objectifs, et la colonne **"Average"** renvoie leurs valeurs moyennes en réalisant 5 exécutions indépendantes sur chaque instance problème.

Instances	SA						ISA					
	Best			Average			Best			Average		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
100_0,2_a	3676	92	415	3638	80,4	393,4	**3687**	99	445	**3687**	99	445
100_0,2_b	3459	83	392	3392	75,4	368,2	**3553**	11	518	**3540**	103,8	511,8
100_0,2_c	3556	73	374	3536	72,4	356,2	**3636**	87	488	**3606**	92,4	471,6
100_0,2_d	3530	80	423	3517	85,6	410,8	**3681**	116	442	**3664**	113,8	457,4
100_0,2_e	3555	80	406	3496	80,8	406,4	**3653**	100	489	**3639**	93,8	448
100_0,5_a	2569	42	257	2642	49,6	297,6	**2776**	68	401	**2698**	62	348
100_0,5_b	2731	43	246	2711	47,2	275,6	**2888**	66	389	**2865**	64,6	382,6
100_0,5_c	**2525**	**51**	**263**	**2515**	**46,8**	**243,6**	2522	64	375	2502	60	328,7
100_0,5_d	2551	51	262	2504	46	256,4	**2685**	65	345	**2630**	73,2	386,8
100_0,5_e	2892	49	292	2833	53,2	324,6	**2963**	56	343	**2948**	62,8	416,2
100_0,8_a	2845	47	279	2823	44,6	290,4	**2886**	68	487	**2850**	62,2	491,4
100_0,8_b	2735	41	237	2781	42,6	266,2	**3024**	56	443	**2990**	59	416,8
100_0,8_c	2810	43	241	2795	40,6	245	**2832**	55	441	**2814**	52	400,8
100_0,8_d	2715	45	233	2698	45	242,4	**2778**	65	575	**2753**	58	431
100_0,8_e	2683	47	218	2619	42,2	243,6	**2748**	58	501	**2740**	61,2	415,8

TABLEAU 3.1 – Impact du mécanisme de refroidissement

Sur l'ensemble des résultats obtenus nous constatons que sur le premier objectif, l'approche **ISA** a obtenu de meilleurs résultats pour 14 instances sur les 15 instances testées avec un taux d'amélioration moyen de $3,67\%$ et de $10,57\%$ comme taux d'amélioration maximum. Pour l'instance (100-0.5-c) le résultat obtenu par **SA** est meilleur que celui obtenu par **ISA**, avec une amélioration ne dépassant pas 0.12%. De plus, pour les valeurs moyennes du premier objectif, les valeurs renvoyées par l'algorithme **ISA** sont globalement meilleures, 14 meilleures valeurs sur 15 instances, avec un taux d'amélioration moyen de $3,47\%$ et un taux maximum d'amélioration de $7,51\%$. En revanche pour l'instance (100-0.5-c) l'approche **SA** a obtenu un meilleur résultat avec une amélioration de seulement 0.55%. Pour les deux autres objectifs, l'approche **ISA** obtient des valeurs moins bonnes que celles obtenues par l'approche **SA**. Cela est dû au fait de colorer plus de sommets, afin de favoriser le premier objectif qui est plus dominant dans l'objectif agrégé qui est meilleur par un taux moyen de 3% et 8% du taux d'amélioration maximum.

3.3.3 Impact de la technique de coloration proposée

Afin de mesurer l'efficacité de la technique de coloration proposée, nous l'avons comparé à la technique R^{SD} connue comme meilleure procédure de coloration proposée dans les travaux de Thevenin et al. (2013, 2018b). Pour ce faire, nous utilisé deux approches recuit simulé identiques sur tous les mécanismes sauf pour la technique de coloration, les deux approches sont nommés ISA et ISA^{SD}, respectivement. A cet effet, nous avons repris les 15 instances de taille 100 sommets utilisées précédemment (section 4.3.3). Les résultats obtenus sont résumés dans la Table 3.2, où la colonne **"Best"** contient la meilleure valeur des trois objectifs, et la colonne **"Average"** renvoie leurs valeurs moyennes des 5 exécutions indépendantes sur la même instance.

Instances	ISA^{SD}						ISA					
	Best			Average			Best			Average		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
100_0,2_a	3615	97	458	3512,6	82,2	370,4	**3687**	**99**	**445**	3687	99	445
100_0,2_b	3517	89	446	3418,4	93,6	456,2	**3553**	**111**	**518**	3540,4	103,8	511,8
100_0,2_c	3550	82	433	3533	81,2	415,8	**3636**	**87**	**488**	3605,8	92,4	471,6
100_0,2_d	3604	82	360	3554,6	86,2	380,2	**3681**	**116**	**442**	3664,4	113,8	457,4
100_0,2_e	3595	70	341	3576,6	71,4	358,4	**3653**	**100**	**489**	3639	93,8	448
100_0,5_a	2763	51	267	2642,4	51,6	295	**2776**	**68**	**401**	2697,6	62	348
100_0,5_b	2769	54	363	2749,8	51	314,2	**2888**	**66**	**389**	2864,6	64,6	382,6
100_0,5_c	2468	51	331	2464	50	300,4	**2522**	**64**	**375**	2501,67	60	328,67
100_0,5_d	2578	55	307	2597,4	56,2	293	**2685**	**65**	**345**	2630,2	73,2	386,8
100_0,5_e	2771	51	236	2842,6	52,4	290	**2963**	**56**	**343**	2947,6	62,8	416,2
100_0,8_a	2832	59	378	2791,8	48,4	300	**2886**	**68**	**487**	2849,8	62,2	491,4
100_0,8_b	3021	40	274	2934,6	46,4	344,2	**3024**	**56**	**443**	2989,5	59	416,75
100_0,8_c	2805	45	250	2766	47,4	333,4	**2832**	**55**	**441**	2813,6	52	400,8
100_0,8_d	2760	46	316	2731,6	47,4	291,2	**2778**	**65**	**575**	2753	58	431
100_0,8_e	2736	42	234	2684,8	52,8	342,4	**2748**	**58**	**501**	2740,2	61,2	415,8

TABLEAU 3.2 – Impact de la technique de coloration proposée

En analysant les résultats obtenus on peut constater clairement que l'approche **ISA** obtient des résultats largement meilleurs pour le premier objectif, et ce pour toutes les instances testées, et ce avec un taux d'amélioration moyen de $2,45\%$ et $4,96\%$ de taux d'amélioration maximum. Cependant, ISA^{SD} a re-

tourné de meilleurs résultats pour les deux autres objectifs sans influencer la dominance de l'objectif agrégé qui est meilleur par un taux moyen de $2,45\%$ et $4,97\%$ du taux d'amélioration maximum.

3.3.4 Comparaison & résultats

Dans cette section, nous présentons une comparaison de notre approche **ISA** avec quelques approches les plus connues dans la littérature pour la résolution de problème d'ordonnancement étudié. Comme nous avons cité antérieurement Thevenin et al. (2018b) ont proposé une méthode exacte basée sur un modèle mathématique en nombres mixtes entiers pour la résolution de petites instances en utilisant le solveur ILP. Ils ont également proposé cinq variantes d'approches de recherche tabou avec différentes techniques de voisinage. D'après les expérimentations réalisées sur les approches proposées, les trois variantes qui ont obtenu de meilleurs résultats sont utilisées dans la comparaison, à savoir (TS^{DIV}, TS^{E1} et TS^{E2}).

Les résultats expérimentaux sont résumés dans les Tables 3.3, 3.4, 3.5, 3.6 et 3.7, décrivant les résultats des instances de 10, 25, 50, 100 et 500 respectivement. Dans les Tables 3.3, 3.4 et 3.5, la colonne 1 fait référence au nom de l'instance et à sa taille correspondante. Les colonnes 2, 3 et 4 rapportent les meilleurs résultats obtenus des travaux du Thevenin et al. (2018b) pour les trois objectifs f_1, f_2 et f_3 respectivement. Les colonnes 17, 18 et 19 rapportent les meilleurs résultats obtenus par **ISA**. Les colonnes 5 à 16 correspondent à l'écart entre la performance moyenne de chaque approche et la meilleure valeur de l'objectif correspondant, à savoir ILP_{gap} pour la résolution Cplex, TS^{DIV}_{gap}, TS^{E1}_{gap} etTS^{E2}_{gap} pour les approches basées sur la recherche tabou. Les colonnes 17 à 19 correspondent à la différence entre la performance moyenne d'ISA_{gap} et la meilleure valeur de l'objectif correspondant. Cependant, les Tables 3.6 et 3.7 rapporte les mêmes données sauf ILP_{gap}.

Dans ce qui suit, nous analysons les performances de l'algorithme ISA pour optimiser l'objectif agrégé. Pour les petites instances, comme on peut le voir

dans la Table 3.3, l'algorithme ISA proposé et les deux approches ILP, TS^{E1} renvoient des solutions optimales pour toutes les instances de 10 sommets sur les 5 exécutions. Cependant, TS^{DIV}_{gap} et TS^{E2}_{gap} ne renvoient que 1 et 7 solutions optimales respectivement. De plus, pour les instances de 25 sommets, ISA renvoie 13 meilleures solutions sur les 15 instances et a amélioré la meilleure solution connue BKS pour les instances identifiées par une valeur négative sur ISA_{gap} dans la Table 3.4.

A partir de ces valeurs négatives, il est facile de voir que ISA est meilleur en termes de valeur moyenne des solutions. De plus, pour les instances moyennes, ISA renvoie 13 meilleures solutions sur les 15 instances de 50 sommets, et 12 meilleures valeurs en termes de solutions moyennes, dépassant de loin les valeurs des autres approches comparées comme résumé dans la Table 3.5. Concernant les grandes instances rapportées dans les Tables 3.6 et 3.7 (instances de taille 100 et 500 sommets), ISA maintient ses performances en donnant de meilleures solutions pour toutes les instances de 100 sommets et 7 meilleures solutions sur 15 testées pour les instances de 500 sommets. En résumé, sur les 75 instances testées, notre approche ISA renvoie 41 meilleures solutions (soit 55%), et donne la même solution pour 23 instances (en comparaison avec les approches ILP_{gap}, TS^{DIV}_{gap}, TS^{E1}_{gap} et TS^{E2}_{gap}.

Un autre résultat notable réside dans l'écart moyen (en pourcentage) entre la valeur moyenne de chacun des trois objectifs et leur valeur correspondante du BKS donnée dans Thevenin et al. (2018b). Cette métrique présentée dans les Tables 3.8, 3.9, explique que plus l'écart est petit, meilleure est la solution. En effet, à partir de la valeur de cette métrique, nous pouvons voir que ISA est meilleur sur l'ensemble de toutes les solutions de toutes les instances testées, sauf une seule avec 500 sommets et une densité de 0.2, où TS^{DIV} est le meilleur et suivi de notre approche.

Inst.	Literature TS															ISA					
	BKS			ILP_{gap}			TS^{Div}_{gap}			TS^{E1}_{gap}			TS^{E2}_{gap}			ISA_{Best}			ISA_{gap}		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
10_0,2_a*	300	5	28	0	0	0	0	0	0	0	0	0	0	0	0	**300**	**5**	**28**	0	0	0
10_0,2_b*	286	9	35	0	0	0	0	2.2	3.4	0	0	0	0	0	0	**286**	**9**	**35**	0	0	0
10_0,2_c*	537	8	38	0	0	0	0	3.8	7.9	0	0	0	0	0	0	**537**	**8**	**38**	0	0	0
10_0,2_d*	348	9	34	0	0	0	0	1	5.3	0	0	0	0	0	0	**348**	**9**	**34**	0	0	0
10_0,2_e*	339	9	41	0	0	0	0	4	2.9	0	0	0	0	2.2	2	**339**	**9**	**41**	0	0	0
10_0,5_a*	321	9	42	0	0	0	0	24.8	53.8	0	0	0	0	11.1	4.3	**321**	**9**	**42**	0	0	0
10_0,5_b*	528	9	47	0	0	0	0	10.1	22.6	0	0	0	0	2.2	0.4	**528**	**9**	**47**	0	0	0
10_0,5_c*	194	6	21	0	0	0	0	0	0	0	0	0	0	0	0	**194**	**6**	**21**	0	0	0
10_0,5_d*	603	8	50	0	0	0	0	1.3	0	0	0	0	0	2.5	0.4	**603**	**8**	**50**	0	0	0
10_0,5_e*	402	9	44	0	0	0	0	23.2	25.9	0	0	0	0	24.4	18.2	**402**	**9**	**44**	0	0	0
10_0,8_a*	434	9	40	0	0	0	0	32.5	79.5	0	0	0	0	8.9	10.5	**434**	**9**	**40**	0	0	0
10_0,8_b*	566	7	49	0	0	0	0	76.6	147.3	0	0	0	0	37.1	38	**566**	**7**	**49**	0	0	0
10_0,8_c*	209	7	19	0	0	0	0	0	0	0	0	0	0	0	0	**209**	**7**	**19**	0	0	0
10_0,8_d*	453	7	45	0	0	0	0	15.6	43.6	0	0	0	0	17.1	17.8	**453**	**7**	**45**	0	0	0
10_0,8_e*	365	5	26	0	0	0	0	2	1.5	0	0	0	0	0	0	**365**	**5**	**26**	0	0	0

TABLEAU 3.3 – Résultats des instances mono-objectif de taille 10 sommets

Inst.	Literature TS															ISA					
	BKS			ILP_{gap}			TS^{Div}_{gap}			TS^{E1}_{gap}			TS^{E2}_{gap}			ISA_{Best}			ISA_{gap}		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
25_0,2_a	1054	21	107	0	0	0	0	46.9	71.6	0	0	0.7	0	12.4	11.4	**1054**	**21**	**107**	0	0	0,56
25_0,2_b*	1120	21	95	0	0	0	0	31.2	37.7	0	0	0	0	14.3	12.2	**1120**	**21**	**95**	0	0	0
25_0,2_c*	1179	21	112	0	0	0	0	62.3	71.1	0	5.7	8.9	0	40	25.2	**1179**	**21**	**112**	0	0	0
25_0,2_d*	1070	18	95	0	0	0	0	32.8	32.6	0	0	0	0	17.8	15.6	**1070**	**18**	**95**	0	0	0
25_0,2_e	907	22	97	0	0	0	0	19.4	31.1	0	0	0	0	10	12.6	907	23	101	0	4,35	3,96
25_0,5_a*	1096	18	109	0	0	0	0	152.1	232.8	0	15.6	20.9	0	78.9	113.9	**1096**	**18**	**109**	0	0	0
25_0,5_b	1419	21	139	0	0	0	0	133.3	166.8	0	11.4	15.3	0	61	48.5	**1419**	**21**	**138**	0	0	-0,72
25_0,5_c	1389	26	181	0	46.2	18.2	0	93.7	76.2	0	10.8	3.4	0	65.4	28.5	**1389**	**25**	**158**	0	-4	-14,56
25_0,5_d	1481	22	147	2.5	0	0	0.2	31.8	131	0	39.1	35.1	0	99.1	50.6	1481	23	151	0	7,56	4,92
25_0,5_e	1366	24	118	0	0	0	0	62.1	133.6	0	0	0	0	32.5	30.5	**1366**	**24**	**118**	0	3,23	3,75
25_0,8_a	1126	27	173	0	0	0	0	123.6	240.6	0	22.2	24.3	0	75.6	74.7	**1126**	**23**	**117**	0	-17,39	-47,86
25_0,8_b*	1636	23	142	0	13	27.5	0	145.1	317.5	0	4.3	5.9	0	101.7	123.7	**1636**	**23**	**142**	0	0	0
25_0,8_c	961	17	99	0	0	0	0	146.5	132.7	0	48.2	36.2	0	122.4	74.1	**961**	**15**	**87**	0	-13,33	-13,79
25_0,8_d	1026	31	234	1.2	0	0	0	101.5	145.6	0	16.8	8.5	0	96.8	65.9	**1026**	**23**	**123**	0	-34,78	-90,24
25_0,8_e	1299	21	126	0	28.6	46.8	0	186.1	368.4	0	47.6	86	0	151.4	181.4	**1299**	**20**	**116**	0	-5	-8,62

TABLEAU 3.4 – Résultats des instances mono-objectif de taille 25 sommets

Inst.	Literature TS															ISA					
	BKS			ILP_{gap}			TS^{Div}_{gap}			TS^{E1}_{gap}			TS^{E2}_{gap}			ISA_{Best}			ISA_{gap}		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
50_0,2_a	2360	61	304	-0.4	18	0	0.2	47.4	49	0.4	6.6	1.8	0.2	47.4	49	**2368**	**57**	**319**	-0,08	2,87	11,06
50_0,2_b	2009	56	266	0	3.6	24.8	0	26.6	39.6	0.1	-6.4	-2.4	0	26.6	39.6	**2009**	**46**	**239**	0	-11,55	3,76
50_0,2_c	2186	46	251	0	4.3	1.2	0.2	67.8	68.9	0.3	2.6	4.9	0.2	67.8	68.9	2186	47	251	0	4,96	2,11
50_0,2_d	2274	78	344	1.1	0	0	0.8	16.4	16.9	1.5	-29.2	-25.8	0.8	16.4	16.9	**2282**	**73**	**346**	0,25	-19,63	-10,75
50_0,2_e	2279	76	365	0.5	0	0	0.9	9.5	8.7	1.3	-18.4	-15.3	0.9	9.5	8.7	**2300**	**60**	**297**	0,59	-38,18	-31,29
50_0,5_a	2000	62	312	13.4	45.2	57.1	1.6	27.4	60.7	0.8	-19.7	-13.3	1.6	27.4	60.7	**2020**	**33**	**206**	1,1	-72,22	-39,04
50_0,5_b	1989	73	340	22	20.5	20.9	0.8	20	67	2.2	-18.1	4.1	0.8	20	67	**2000**	**42**	**242**	1	-77,18	-41,9
50_0,5_c	1856	60	266	1.6	31.7	36.1	1.6	34.3	75.9	0.9	-8.3	-0.8	1.6	34.3	75.9	**1856**	**33**	**177**	0	-62,16	-19,71
50_0,5_d	2004	116	556	8.5	0	0	0.2	-18.9	12.5	0.6	-41	-34.3	0.2	-18.9	12.5	**2033**	**53**	**272**	-0,88	-115,61	-105,32
50_0,5_e	1969	61	286	10.4	62.3	4.4	0.8	23.9	54.5	0.5	0.3	7.5	0.8	23.9	54.5	**1983**	**43**	**234**	0,23	-63,98	-39,92
50_0,8_a	2205	46	317	17.9	0	0	0.7	145.8	220.4	0.3	106.5	78.9	0.7	145.8	220.4	**2236**	**36**	**238**	-1,35	-25	-31,64
50_0,8_b	2451	76	676	17	0	0	1.3	66.6	54.8	2	21.3	-5	1.3	66.6	54.8	**2451**	**39**	**345**	0	-79,25	-84,6
50_0,8_c	2414	51	532	16.5	0	0	0.3	139.9	81.2	0.8	61.6	16.2	0.3	139.9	81.2	**2433**	**40**	**333**	-0,78	-40,11	-84,85
50_0,8_d	1561	54	387	10.2	0	0	0.4	109.8	175.8	1.2	27.4	39.6	0.4	109.8	175.8	**1561**	**45**	**347**	1,01	-33	-35,69
50_0,8_e				3.7	27.9	4.8	0.5	57.1	23.2	0.8	19.1	-13.3	0.5	57.1	23.2	1963	43	329			

TABLEAU 3.5 – Résultats des instances mono-objectif de taille 50 sommets

Inst.	Literature TS												ISA					
	BKS			TS^{Div}_{gap}			TS^{E1}_{gap}			TS^{E2}_{gap}			ISA_{Best}			ISA_{gap}		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
100_2_a	3651	106	491	1.2	63.8	32	1.3	-6	-9	1	18.5	1.3	**3687**	**99**	**445**	-0,98	-7,07	-10,34
100_2_b	3533	103	450	1.1	85	54.4	0.9	3.3	5.3	2.1	8.5	5.6	**3553**	**111**	**518**	-0,21	0,77	12,08
100_2_c	3614	206	713	1.3	-16	-3.6	2.2	-55.5	-38	2.4	-51.5	-36.3	**3636**	**87**	**488**	0,23	-122,94	-51,19
100_2_d	3670	124	508	0.5	36.5	36.6	1.2	-12.7	-3.4	0.9	2.3	5.2	**3681**	**116**	**442**	0,15	-8,96	-11,06
100_2_e	3632	116	495	1.3	52.6	40.1	0.8	-7.4	-7.9	0.8	3.6	-4.5	**3653**	**100**	**489**	-0,19	-23,67	-10,49
100_5_a	2752	128	637	1.6	6.9	4.1	2.6	-40.9	-40.1	2.5	-28.8	-30.6	**2776**	**68**	**401**	2,02	-106,45	-83,05
100_5_b	2850	173	672	3.7	-17.7	0.8	3.1	-52.3	-36.4	4.1	-43.6	-28.8	**2888**	**66**	**389**	-0,51	-167,8	-75,64
100_5_c	2504	87	489	0.7	76.1	49.9	1.2	2.5	-18.6	1.2	20.5	0.6	**2522**	**64**	**375**	0,09	-45	-48,78
100_5_d	2652	120	617	2.6	40.2	30.7	2.9	-18.3	-20.8	3.1	-9.3	-10.9	**2685**	**65**	**345**	0,83	-63,93	-59,51
100_5_e	2959	130	625	2.4	12.8	8.4	3.2	-32.9	-30.3	3.3	-28.5	-27.6	**2963**	**56**	**343**	0,39	-107,01	-50,17
100_8_a	2878	172	1348	1.9	1.9	0.4	2.9	-25.6	-39.4	2	-12.7	-26.6	**2886**	**68**	**487**	0,99	-176,53	-174,32
100_8_b	2962	130	879	2.3	29.5	41.6	3.2	-0.3	-3.2	2.4	6.8	9.9	**3024**	**56**	**443**	-0,92	-120,34	-110,92
100_8_c	2770	123	851	1.4	1.1	5.9	1	-0.5	-9.5	1.9	-11.2	-13.4	**2832**	**55**	**441**	-1,55	-136,54	-112,33
100_8_d	2763	152	1196	2.1	10.8	3.3	1.7	-22	-35.9	1.8	-19.2	-30	**2778**	**65**	**575**	0,36	-162,07	-177,49
100_8_e	2710	166	1185	0.6	-2.7	-0.7	1.9	-28.1	-37.6	1.4	-16.1	-24.5	**2748**	**58**	**501**	-1,1	-171,24	-184,99

TABLEAU 3.6 – Résultats des instances mono-objectif de taille 100 sommets

Inst.	Literature TS												ISA					
	BKS			TS^{Div}_{gap}			TS^{E1}_{gap}			TS^{E2}_{gap}			ISA_{Best}			ISA_{gap}		
	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
500_2_a	21608	1068	11274	0.8	18.2	11.7	4.8	-16.8	-35.5	4.4	-14	-34.7	20820	602	5950	4,99	-40,9	-47,24
500_2_b	27414	599	5694	1.7	2.5	11.4	1.7	-2.4	-7.1	0.7	-2.8	-11	**27884**	**1454**	**18389**	-1,71	133,29	68,18
500_2_c	25331	1664	19339	0.7	-3.7	-1.3	4.4	-35.2	-42.3	4.4	-46.4	-55.6	24796	990	10696	2,2	-41,95	-76,71
500_2_d	26026	1524	19716	0.8	-6.1	-2.8	5.8	-56.7	-64.9	6.1	-61.3	-71.2	25581	1224	13620	1,71	-19,69	-44,76
500_2_e	26064	1650	20079	0.8	-3.6	-5.9	6.4	-41.9	-52.7	5.7	-48	-57.6	25378	985	10378	2,76	-40,39	-87,76
500_5_a	23875	1074	31257	1.6	-2	10.5	4.9	-39.2	-60	5.3	-42.7	-64.1	23521	795	17749	1,78	-24,13	-71,37
500_5_b	13589	887	12623	1.2	-11.9	-13.4	4.2	-42.9	-53.3	3.4	-30	-38.4	13240	316	4263	3,42	-64,33	-196,62
500_5_c	20456	1131	24217	1.4	-8.4	-8.3	4.5	-40.9	-52.3	4.7	-36.9	-51.3	19881	549	11002	2,81	-51,46	-120,11
500_5_d	26030	988	29372	1.4	-8.2	-14.1	3.3	-32.7	-54.9	2.8	-25.4	-43.9	**26370**	**1005**	**24350**	-1,31	1,72	-20,62
500_5_e	26781	931	23974	1.3	-18.5	-26.8	1.8	-33	-58	1.4	-32.9	-60.2	**27160**	**1180**	**31467**	-1,42	26,75	23,81
500_8_a	18937	615	20056	1.6	4.4	7.4	3.2	-3.6	-18.2	3.2	-9.2	-26.1	18485	359	9726	2,39	-41,63	-106,21
500_8_b	25953	752	23198	1	-8	-13.4	1.4	-19.4	-30.2	1	-17.3	-34.3	**26333**	**817**	**31576**	-1,46	8,64	26,53
500_8_c	24253	823	34679	2.2	-25.9	-44.9	2	-29.6	-54.3	1.4	-32.4	-57.3	**24563**	**739**	**28627**	-1,28	-10,21	-21,14
500_8_d	23701	710	27318	2.1	-12.4	-20	3.5	-26	-47.4	3.1	-24.7	-49.6	**23823**	**689**	**27963**	-0,51	0,08	-14,56
500_8_e	27087	618	15833	1.2	3	27.8	1.1	4	20.3	0.5	1.5	10	**27684**	**968**	**43559**	-2,16	50,16	62,28

TABLEAU 3.7 – Résultats des instances mono-objectif de taille 500 sommets

n	d	ILP	TS^{Div}	TS^{E1}	TS^{E2}	ISA
10	0.2	(0 ; 0 ; 0)	(0 ; 2.2 ; 3.9)	(0 ; 0 ; 0)	(0 ; 0.4 ; 0.4)	**(0 ; 0 ; 0)**
	0.5	(0 ; 0 ; 0)	(0 ; 11.9 ; 20.5)	(0 ; 0 ; 0)	(0 ; 8.1 ; 4.7)	**(0 ; 0 ; 0)**
	0.8	(0 ; 0 ; 0)	(0 ; 25.3 ; 54.4)	(0 ; 0 ; 0)	(0 ; 12.6 ; 13.2)	**(0 ; 0 ; 0)**
	Avg	(0 ; 0 ; 0)	(0 ; 13.1 ; 26.3)	(0 ; 0 ; 0)	(0 ; 7 ; 6.1)	**(0 ; 0 ; 0)**
25	0.2	(0 ; 0 ; 0)	(0 ; 38.5 ; 48.8)	(0 ; 1.1 ; 1.9)	(0 ; 18.9 ; 15.4)	**(0 ; 0.91 ; 0.9)**
	0.5	(0.5 ; 9.2 ; 3.6)	(0 ; 114.6 ; 148.1)	(0 ; 15.4 ; 14.9)	(0 ; 67.4 ; 54.4)	**(0 ; 1.53 ;-1.32)**
	0.8	(0.2 ; 8.3 ; 14.9)	(0 ; 140.6 ; 240.9)	(0 ; 27.8 ; 32.2)	(0 ; 109.6 ; 104)	**(0 ; -11.43 ; -32.1)**
	Avg	(0.2 ; 5.9 ; 6.2)	(0 ; 97.9 ; 146)	(0 ; 14.8 ; 16.3)	(0 ; 65.3 ; 57.9)	**(0 ; -3 ; -10.84)**
50	0.2	(0.2 ; 5.2 ; 5.2)	(0.4 ; 33.5 ; 36.6)	(0.7 ; -9 ; -7.3)	(0.6 ; 11.8 ; 1.5)	**(0.15 ; -9.25 ; -4.75)**
	0.5	(11.2 ; 31.9 ; 31.7)	(1 ; 17.4 ; 54.1)	(1 ; -17.4 ; -7.4)	(0.8 ; -5 ; -0.2)	**(0.28 ; -43.29 ; -49.18)**
	0.8	(13.1 ; 5.6 ; 1)	(0.6 ; 103.8 ; 111.1)	(1 ; 47.2 ; 23.2)	(0.8 ; 78.9 ; 38.8)	**(-0.29 ; -29.41 ; -59.2)**
	Avg	(8.2 ; 14.2 ; 12.6)	(0.7 ; 51.6 ; 67.3)	(0.9 ; 6.9 ; 2.8)	(0.7 ; 28.6 ; 13.3)	**(0.05 ; -27.32 ; -37.71)**
Avg.		(2.8 ; 6.7 ; 6.3)	(0.2 ; 54.2 ; 79.8)	(0.3 ; 7.2 ; 6.4)	(0.2 ; 33.6 ; 25.8)	**(0.02 ; -10.1 ; -16.18)**

TABLEAU 3.8 – Récapitulatif des résultats de petites instances

n	d	TS^{Div}	TS^{E1}	TS^{E2}	ISA
100	0.2	(1.1 ; 44.4 ; 31.9)	(1.3 ; -15.7 ; -10.6)	(1.4 ; -3.7 ; -5.8)	**(-0,2 ; -17,67 ; -14,01)**
	0.5	(2.2 ; 23.6 ; 18.8)	(2.6 ; -28.4 ; -29.2)	(2.9 ; -17.9 ; -19.5)	**(0,55 ; -47,19 ; -63,43)**
	0.8	(1.7 ; 8.1 ; 10.1)	(2.1 ; -15.3 ; -25.1)	(1.9 ; -10.5 ; -16.9)	**(-0,46 ; -60,23 ; -152,01)**
	Avg	(1.6 ; 25.4 ; 20.3)	(2 ; -19.8 ; -21.7)	(2.1 ; -10.7 ; -14)	**(-0,46 ; -60,23 ; -152,01)**
500	0.2	(1 ; 1.5 ; 2.6)	(4.6 ; -30.6 ; -40.5)	(4.3 ; -34.5 ; -46)	(1,99 ; -1,93 ; -37,66)
	0.5	(1.4 ; -9.8 ; -14.6)	(3.7 ; -37.7 ; -55.7)	(3.5 ; -33.6 ; -51.6)	**(1,06 ; -22,29 ; -76,98)**
	0.8	(1.6 ; -7.8 ; -8.6)	(2.2 ; -14.9 ; -26)	(1.8 ; -16.4 ; -31.5)	**(-0,6 ; 1,41 ;-10,62)**
	Avg	(1.3 ; -5.4 ; -6.9)	(3.5 ; -27.8 ; -40.7)	(3.2 ; -28.2 ; -43)	**(0,81 ; -7,6 ; -41,75)**
Avg.		(1.5 ; 10 ; 2.3)	(2.8 ; -23.8 ; -31.2)	(2.7 ; -19.45 ; -29)	**(0,18 ; -33,9 ; -96,9)**

TABLEAU 3.9 – Récapitulatif des résultats de grandes instances

3.4 Conclusion

Dans ce chapitre nous avons traité un problème d'ordonnancement d'un ensemble de jobs préemptifs, qui doivent être exécutées avant une échéance globale prédéfinie, sur un ensemble de machines parallèles. Le problème considère trois objectifs à optimiser avec un ordre lexicographique modélisant plusieurs cas pratiques.

Ce problème d'ordonnancement est modélisé par multicoloration de graphes où chaque sommet du graphe représentant un job, est affecté par un ensemble de couleurs tout en respectant toutes les contraintes du problème. Pour résoudre ce problème NP-difficile, un algorithme de recuit simulé amélioré avec un mécanisme de refroidissement est proposé. Ainsi qu'une technique de coloration basée sur la minimisation du gain perdu lors de la coloration des sommets, dans le but de renforcer l'efficacité de l'algorithme proposé.

Afin de valider les performances de l'approche proposée ISA, des tests sur 75 instances de référence bien connues ont été réalisés. Ces tests ont monté une performance plus élevée relativement beaucoup plus puissante par rapport à celles des autres méthodes de l'état de l'art sur de nombreux instances.

Chapitre 4

Multicoloration de graphes pour l'ordonnancement bi-objectif : Approches de résolution

Sommaire

4.1 Introduction

De nombreux secteurs de l'industrie sont concernés par des problèmes complexes à optimiser. Ces problèmes d'optimisation sont rarement à objectif unique. Au contraire, ils ont souvent plusieurs critères ou objectifs contradictoires qui doivent être satisfaits simultanément. De manière similaire, les problèmes d'ordonnancement sont très souvent de nature multi-objectif. Néanmoins la majorité des travaux qui lui sont consacrés le considèrent sous une forme mono-objectif et vise généralement à minimiser la date de fin de l'ordonnancement (”le makespan”).

Dans ce chapitre, nous abordons le problème d'ordonnancement étudié dans le chapitre 2 en considérant deux objectifs, à savoir, la maximisation du gain total des jobs effectués et la minimisation de la somme des dates d'achèvement des jobs (total flow time ou total completion time). Pour ce faire, nous commençons d'abord par une présentation des diverses notions essentielles de l'optimisation multi-objectif.

4.2 Optimisation multi-objectif

Dans la plupart des cas pratiques des problèmes d'optimisation combinatoire, la prise en compte d'un critère unique est malheureusement insuffisante. En effet, on rencontre rarement ce cas de figure dans la pratique, où pour la plupart des problèmes il est nécessaire d'optimiser simultanément plusieurs critères contradictoires. Dans ce type de problèmes, on aura recours à l'optimisation multi-objectif. Pour les résoudre il ne s'agit plus de trouver une solution optimale mais une multitude (ensemble) de solutions du fait que certains critères sont conflictuels, cet ensemble de solutions représente un compromis acceptable entre les différents objectifs souvent contradictoires.

4.2.1 Problème d'optimisation multi-objectif

Un problème d'optimisation multi-objectif est un problème combinatoire, dont on recherche l'action qui satisfait un ensemble de contraintes tout en optimisant un vecteur de fonctions objectives c.-à-d. optimiser plusieurs objectifs contradictoires ou conflictuels. Ainsi, un problème d'optimisation multi-objectif est un problème de la forme suivante :

$$(\mathrm{P}) = \begin{cases} \mathrm{M}in\ \mathrm{Z}(x) = (z_1(x),\ z_2(x), ...,\ z_k(x)) \quad k \geq 2 \\ \text{sous contraintes : } x \in \mathrm{S} \end{cases} \tag{4.1}$$

Avec :

$\mathrm{X} = (x_1, x_2, ..., x_n)$ **:** est le vecteur de n variables de décision.

$\mathrm{Z}(x) = (z_1(x), z_2(x), ..., z_k(x))$ **:** est le vecteur de k fonctions objectifs.

S **:** est l'ensemble non vide des solutions réalisables du problème.

Soit $\mathrm{Z_S}$ l'ensemble des objectifs réalisables défini par les images des solutions réalisables x dans S, on associe à S l'ensemble $\mathrm{Z}(x)$ des objectifs correspondants à ces composantes $z(x) = (z_1(x),\ z_2(x), ...,\ z_k(x))$ dans R^k tel que $(x \in \mathrm{S})$ et on construit donc le polyèdre $\mathrm{Z_S} = \{z \in \mathrm{R}^k | z = \mathrm{C}x; x \in \mathrm{S}\}$. Si l'ensemble S est un polyèdre convexe alors l'ensemble $\mathrm{Z_S}$ est un polyèdre convexe dans R^k dont les sommets sont des images des sommets de S (Figure 4.1). Sans perte de généralité, nous supposerons dans la suite que nous considérons des problèmes de minimisation.

4.2.2 Théorie d'optimisation au sens de Pareto

4.2.2.1 Optimum de Pareto

Au XIX^{ème} siècle, Vilfredo Pareto (1896), un mathématicien italien, a formulé le concept suivant : ***« Dans un problème d'optimisation multi-objectif, il existe un équilibre tel l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères »***. Cet équilibre est appelé optimum de Pareto, en effet, un point x est dit Pareto-optimal s'il n'est dominé par aucun

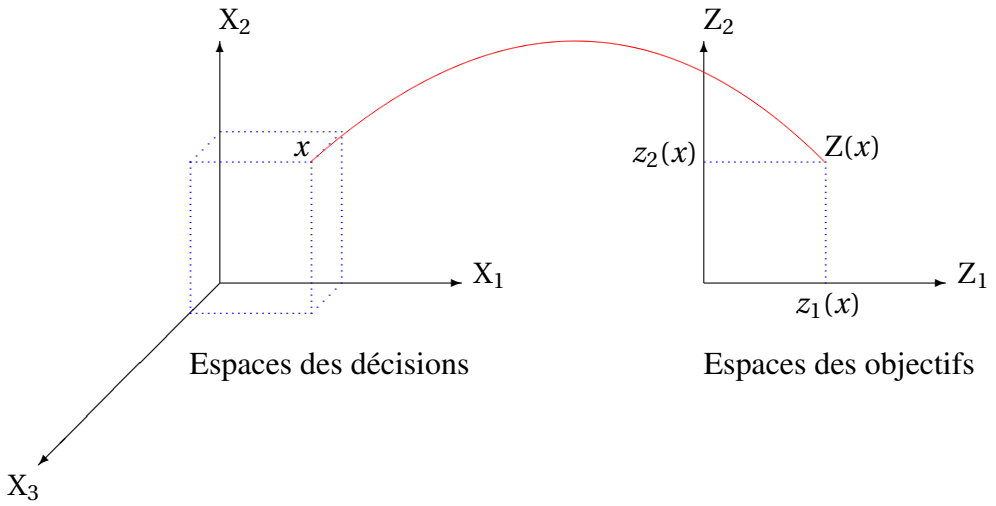

FIGURE 4.1 – Représentation de l'espace des objectifs

autre point appartenant à S, ces points sont également appelés solutions non inférieures ou non dominées.

4.2.2.2 Notion de dominance

Le principe de dominance est utilisé par la majorité des algorithmes d'optimisation multi-objectif pour comparer deux solutions.

Definition 4.2.1 (Principe de dominance)**.** Une solution $x(i)$ domine une autre solution $x(j)$ si les deux conditions suivantes sont vérifiées :

1. $z_k(x^{(i)}) \leq z_k(x^{(j)}) \quad \forall k \in \{1,...,\mathrm{K}\}$;
2. $\exists k \in \{1,...,\mathrm{K}\}$ telque $z_k(x^{(i)}) < z_k(x^{(j)})$.

Si $x^{(i)}$ domine $x^{(j)}$, cette relation est notée $x^{(i)} \prec x^{(j)}$.

D'autres relations qui peuvent être dérivées de la relation de dominance sont énoncées ci-dessous :

Definition 4.2.2 (Dominance faible)**.** Une solution $x^{(i)}$ domine strictement une solution $x^{(j)}$ si et seulement si : $z_k(x^{(i)}) < z_k(x^{(j)}) \quad \forall k \in \{1,...,\mathrm{K}\}$. Cette relation est notée $x^{(i)} \prec\prec x^{(j)}$.

Definition 4.2.3 (ϵ-dominance)**.** Une solution $x^{(i)}$ ϵ-domine une solution $x^{(j)}$ si et seulement si : $z_k(x^{(i)}) \leq \epsilon.z_k(x^{(j)}) \quad \forall k \in \{1, ..., \mathrm{K}\}$. Cette relation est notée $x^{(i)} \preceq_\epsilon x^{(j)}$.

Definition 4.2.4 (ϵ-dominance additive)**.** Une solution $x^{(i)}$ ϵ-domine additivement une solution $x^{(j)}$ si et seulement si : $z_k(x^{(i)}) \leq \epsilon + z_k(x^{(j)}) \quad \forall k \in \{1, ..., \mathrm{K}\}$. Cette relation est notée $x^{(i)} \preceq_{\epsilon+} x^{(j)}$.

4.2.2.3 Frontière de Pareto

Pour un ensemble de solutions fini, toutes les solutions peuvent être comparées deux à deux selon le principe de dominance, et nous pouvons déduire quelle solution domine l'autre. A la fin, nous obtenons un ensemble où aucune des solutions ne domine l'autre, cet ensemble est appelé ensemble des solutions non dominées ou bien ensemble des solutions Pareto-optimales.

Definition 4.2.5 (Ensemble des solutions non-dominées)**.** Soit P un ensemble de solutions, l'ensemble des solutions non dominées $\mathrm{P}' \in \mathrm{P}$ est l'ensemble des solutions non-dominées par aucun autre membre de l'ensemble P.

Si l'ensemble P représente la totalité de l'espace de recherche S, l'ensemble des solutions non-dominées P' est appelé ensemble pareto-optimal dans l'espace des décisions ou front de pareto dans l'espace des objectifs.

Definition 4.2.6 (Ensemble pareto-optimal)**.** L'ensemble pareto-optimal est l'ensemble des solutions non-dominées de l'espace de recherche faisable S.

On dit qu'une solution $x \in \mathrm{S}$ est efficace, s'il n'existe pas une autre solution $y \in \mathrm{S}$ telle que le vecteur $\mathrm{Z}(y)$ domine le vecteur $\mathrm{Z}(x)$.

4.2.2.4 Points particuliers et la matrice des gains

En vue d'avoir certains points de références permettant de discuter de l'intérêt des solutions trouvées, des points particuliers ont été définis dans l'espace des critères. Ces points peuvent représenter des solutions réalisables ou

non. Nous trouvons parmi ces points le point idéal, le point nadir et le point anti-idéal.

Point Idéal :

Le point Idéal I est un point de l'espace des critères $\mathbb{R}^K$ de coordonnées : $I_k = Z_1^*, Z_2^*, ..., Z_k^*$ où $Z_j^* = \underset{x \in S}{max}\, Z_j(x),\ \forall\, j = 1, 2, ..., k$, pour un problème de maximisation. En général, ce point ne correspond pas à une solution réalisable car si c'était le cas, cela sous-entendrait que les objectifs ne sont pas contradictoires et qu'une solution optimisant un objectif, optimise simultanément tout les autres objectifs, ce qui ramènerait le problème à trouver une seule solution pareto optimale.

Matrice des Gains :

Soit $x^{*(l)}$ une solution optimale du critère Z_i non nécessairement unique. Notons par $Z_{kl} = Z_k(x^{(l)*})$. La matrice carrée G d'ordre K formée des éléments Z_{kl}, est appelée la matrice des gains.

$$G = \begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1k} \\ Z_{21} & Z_{22} & \cdots & Z_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{k1} & Z_{k2} & \cdots & Z_{kk} \end{pmatrix}$$

Les coordonnées du point idéal apparaissent sur la diagonale. Notons que la matrice des gains n'est pas unique. En effet, s'il existe un critère Z_l ayant plus d'une solution optimale, l'évaluation Z_{kl} change suivant la solution optimale considérée.

Le point nadir N :

C'est un point de l'espace des critères R^K ayant pour coordonnées :
$N_k = \underset{x \in eff}{min}\, Z_k(x),\ \forall k = 1, 2, ..., K$, où $eff \in S$ est l'ensemble des solutions efficaces du problème de maximisation.

Remarque 4.2.1. *Le point nadir correspond à la borne inférieur de la surface de Pareto, et non pas dans tout l'espace réalisable. La recherche des coordonnées du vecteur correspondant est un problème (pour le moins aussi difficile à résoudre) que celui de la recherche des solutions*

efficaces. Le point Nadir sert à restreindre l'espace de recherche, et il est utilisé dans certaines méthodes de résolutions.

Une approximation du point nadir est donnée par : $\overline{\mathrm{N}}_k = \underset{l=\overline{1,k}}{min}\, \mathrm{Z}_{kl},\ \forall k = 1,\ 2,\ ...,\ \mathrm{K}$, qui représente le maximum de chaque colonne de la matrice des gains. Il est clair que si la matrice des gains G n'est pas unique, l'approximation du point nadir $\overline{\mathrm{N}}$ ne le sera pas.

Le anti-Idéal :

Le point anti-idéal I est aussi un point de l'espace des critères ayant pour coordonnées : $\mathrm{I}_k = \mathrm{Z}_1^*,\ \mathrm{Z}_2^*, ...,\ \mathrm{Z}_k^*,\ \forall k = 1,\ 2, ...,\ \mathrm{K}$, où $\mathrm{Z}_j^* = \underset{x\in \mathrm{S}}{min} \mathrm{Z}_j(x),\ \forall j = 1,\ 2, ...,\ j$, pour un problème de maximisation.

4.2.3 Résolution des problèmes d'optimisation multi-objectif

Pour la résolution des problèmes d'optimisation multi-objectif des méthodes exactes, ainsi que des heuristiques ont été proposées. Nous présentons, dans ce qui suit, leurs classifications et les principales techniques afin d'avoir une vue plus globale.

4.2.3.1 Classification des méthodes d'optimisation multi-objectif

Dans la littérature, deux classifications principales des méthodes de résolution des problèmes d'optimisation multi-objetif sont distinguées. La première classification adopte un point de vue décideur. La deuxième classe les méthodes suivant leur façon de traiter les fonctions objectifs.
La première classification divise ces méthodes en trois familles selon la manière dont sont combinés ces processus :

- **Les Méthodes a priori :** Dans ces méthodes, le décideur définit le compromis qu'il désire réaliser. Ainsi, il est supposé connaître à priori le poids de chaque objectif, ce qui revient à transformer le problème d'OM en un problème mono-objectif afin de pouvoir utiliser directement les méthodes de recherche.

- **Les Méthodes a posteriori :** Dans ces méthodes, le décideur choisit une solution parmi toutes les solutions de l'ensemble des solutions Pareto-optimales fourni par la méthode d'optimisation. Ces dernières permettent au décideur d'exprimer ses préférences à l'issue du processus d'optimisation, parmi l'ensemble des solutions retenues (idéalement un ensemble de compromis optimaux au sens de Pareto), ce qui lui permet de tirer un profit maximum de l'étape d'optimisation.
- **Les Méthodes interactives :** Dans ces méthodes, il y a une coopération progressive entre la méthode d'optimisation et le décideur. Le décideur affine son choix de compromis au fur et à mesure du déroulement du processus de résolution à partir des connaissances acquises à chaque étape du processus.

On peut trouver des méthodes d'optimisation multi-objectif qui n'entrent pas exclusivement dans une famille. On peut utiliser, par exemple, une méthode à priori en lui fournissant des préférences choisies aléatoirement. Le résultat sera alors un ensemble pareto optimal à cardinalité élevée qui sera présenté au décideur pour qu'il choisisse une solution. Cette combinaison forme alors une méthode à posteriori. Cette classification permet donc seulement d'avoir une idée sur la démarche que l'on devra suivre pour obtenir la solution.
La deuxième classification divise les méthodes de résolution de problèmes d'OM en deux classes :

- La classe de méthodes transformant le problème initial en un problème mono-objectif, ainsi ces méthodes retournent une solution unique, elles utilisent généralement des méthodes d'agrégation.
- La classe de méthodes fournissant un ensemble de solutions pareto optimales : cette classe est aussi divisée en deux sous-classes.

Il existe une autre classification qui tient en compte le principe de domination au sens de pareto, à savoir :

- **Les Méthodes Pareto :** ces méthodes utilisent directement la notion de dominance au sens de Pareto dans la sélection des solutions géné-

rées. Cette idée a été initialement introduite par Goldberg (1989) pour résoudre les problèmes proposés par Schaffer and Grefenstette (1985).

- **Les Méthodes non Pareto :** ces méthodes optimisent séparément les objectifs.

4.2.3.2 Techniques classiques pour l'optimisation multi-Objectif

Les méthodes classiques de résolution d'un problème d'optimisation multi-objectif ont le but de générer l'ensemble Pareto Optimal. Elles reposent généralement sur la transformation du problème multi-objectif initial en un problème d'optimisation mono-objectif (P). Les paramètres de la fonction employée ne sont pas fournis par le décideur mais par l'optimiseur. Plusieurs exécutions de l'optimiseur sont effectuées dans le but de trouver un ensemble de solutions qui approxime l'ensemble optimal de Pareto. Plusieurs méthodes ont été proposées, nous citons pour exemple la méthode **agrégation pondérée**, la méthode $\epsilon-$**Contraintes**, et **la méthode programmation par but**. Ces méthodes vont être présentées avec plus de détails de qui suit.

Technique d'agrégation pondérée :

L'idée de cette méthode la plus populaire et la plus utilisée dans la pratique est très intuitive. Elle consiste à transformer le problème multi-objectif (P) en un problème mono-objectif (P′) en combinant linéairement les différents objectifs. Ainsi, le nouveau problème obtenu consiste à optimiser :

$$(\mathrm{P}') = \begin{cases} \mathrm{F}(x) = \sum_i \lambda_i . f_i(x) \\ \sum_i \lambda_i = 1 \\ \lambda_i \geq 0 \end{cases} \tag{4.2}$$

La résolution du problème (P′) , en faisant varier les valeurs du vecteur poids λ, on peut retrouver l'ensemble de solutions supportées, si le domaine réalisable est convexe. Cependant, elle ne permet pas de trouver les solutions enfermées dans une concavité (les solutions non supportées), ceci est un résultat des deux théorèmes suivantes :

Théorème 4.2.1. *(Théorème de Geoffrion (1968)) Une solution optimale du problème* (P′) *(equation 4.2) est une solution Pareto-Optimale pour le problème multi-objectif* (P) *(equation 4.1) si tous les poids* λ_i *sont strictement positifs.*

Théorème 4.2.2. *Si le problème d'optimisation multi-objectif* (P) *est convexe et* x^* *est une de ses solutions pareto-optimales, il existe un vecteur de poids positifs* $(\lambda_1, ..., \lambda_k)$ *tel que* x^* *est la solution du problème* (P′).

L'approche généralement utilisée consiste à répéter l'exécution de l'algorithme avec des vecteurs poids différents. Les résultats obtenus avec de telles méthodes dépendent fortement des paramètres choisis pour le vecteur de poids λ. Les poids λ_i doivent également être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate.

Technique $\epsilon-$Contraintes :

Cette méthode a été suggérée par Haimes et al. (1971), elle est basée sur la transformation de $K-1$ des K objectifs du problème (P) en contraintes et l'objectif restant qui peut être choisi arbitrairement, est la fonction objectif du problème (P″) qui en résulte :

$$(P'') = \begin{cases} \min\limits_{x \in S} \; f_k(x) \\ f_m(x) \geq \epsilon_m, \quad m = 1, ..., K \quad \text{et} \quad m \neq k. \end{cases} \tag{4.3}$$

Le choix de différentes valeurs pour $\epsilon = (\epsilon_1, ..., \epsilon_{k-1}, \epsilon_{k+1}, ..., \epsilon_K)$ reste l'étape la plus difficile de cette méthode et ceci nécessite une très bonne connaissance à priori des intervalles appropriés pour les valeurs pour tous les objectifs du problème est requise afin de pouvoir générer l'ensemble des solutions pareto optimales.

Technique de programmation par but (Goal Programming) :

Dans cette méthode également appelée target vector optimisation (Coello, 1996), le décideur doit définir un but T_i qu'il désire atteindre pour chaque

objectif f_i. Ces valeurs sont introduites dans la formulation du problème le transformant en un problème uni-objectif. La nouvelle fonction objectif est modifiée de façon à minimiser la norme pondérée qui minimise les déviations par rapport aux buts par exemple. Le problème est formulé comme suit :

$$min \sum_{i}^{k} |f_i(x) - \mathrm{T_I}| \quad \text{avec} \quad i \in \mathrm{S} \tag{4.4}$$

4.2.3.3 Métaheuristiques pour l'optimisation multi-objectif

Les métaheuristiques ont été largement utilisées pour la résolution des problèmes d'optimisation multi-objectif. Ce sont des méthodes qui ne garantissent pas de trouver de manière exacte tout l'ensemble des solutions Pareto, mais une approximation aussi bonne que possible, de cet ensemble. Dans ce qui suit nous présentons les principales métaheuristiques utilisées pour la résolution des problèmes multi-objectif.

Recuit Simulé :

Le premier algorithme multi-objectif basé sur le recuit simulé a été proposé par Serafini (1992). Cet algorithme utilise le schéma standard du recuit simulé avec une seule solution courante. Le résultat de l'algorithme est un ensemble de solutions pareto-optimales contenant toutes les solutions non-dominées générées par l'algorithme. Serafini considère un certain nombre de règles définissant la probabilité d'accepter des nouvelles solutions voisines. Le recuit simulé mono-objectif accepte avec une probabilité égale à 1 les nouvelles solutions meilleures ou égales à la solution courante. Si la nouvelle solution est mauvaise que la solution courante elle est acceptée avec une probabilité inférieure à 1. Dans le cas multi-objectif, trois situations sont possibles en comparant une nouvelle solution x' avec la solution courante x, x' domine x, x' est dominée par x ou bien elles sont mutuellement non-dominées.

Dans le premier cas, il est évident d'accepter x' avec une probabilité égale à 1. Dans le deuxième cas la probabilité d'acceptation doit être inférieure à 1. La question se pose pour la troisième situation quelle doit être la va-

leur de la probabilité d'acceptation. Serafini propose des règles d'acceptation multi-objectif qui traitent différemment la troisième situation. Cesrègles correspondent à certaines fonctions d'agrégation locales.
Ulungu et al. (1999) ont proposé une méthode appelée MOSA. Ils utilisent aussi des règles d'acceptation multi-objectif. MOSA utilise un nombre de vecteurs poids prédéfinis, chaque vecteur est associé à un processus de recuit indépendant. Chaque processus commence d'une solution aléatoire ou d'une solution construite par une heuristique spécialisée. Ensuite, la solution réalise des mouvements qui sont acceptés avec une probabilité définie par une règle d'acceptation. Le résultat de l'algorithme est un ensemble de solutions Pareto-optimales contenant toutes les solutions non-dominées générées par tous les processus de recuit. Ainsi, ces processus, qui travaillent indépendamment, coopèrent dans la génération d'un ensemble commun de solutions pareto optimales. Czyzak and Jaszkiewicz (1998) ont proposé la méthode "Pareto Simulated Annealing". Cette méthode utilise des fonctions scalaires basées sur les probabilités pour l'acceptation des nouvelles solutions voisines. Une population de solutions est utilisée, chacune d'elles explorant l'espace de recherche relativement aux règles du recuit simulé. Les solutions peuvent être traitées comme des agents travaillant indépendamment mais échangeant des informations sur leurs positions. A chaque solution est associée un vecteur poids séparé. La méthode met à jour les vecteurs poids des solutions en construction en utilisant une règle pour assurer une dispersion sur toutes les régions du front pareto. Suppapitnarm and Parks (2001) ont proposé une méthode dans laquelle la probabilité l'acceptation d'une solution dépend du fait qu'elle est ajoutée ou non à l'ensemble des solutions Pareto optimales potentielles. Si elle est ajoutée à cet ensemble, ce qui signifie qu'elle n'est dominée par aucune autre solution trouvée, elle est acceptée pour être la solution courante avec une probabilité égale à 1. Sinon, une règle d'acceptation multi-objectif est utilisée.

Recherche Tabou :

Grandibleux et al. (1997) ont proposé une version multi-objectif de la re-

cherche tabou, cette méthode utilise des fonctions pondérées scalaires dont les poids sont changés périodiquement. La modification du vecteur poids dégrade les poids des objectifs qui ont été nettement améliorés. Deux listes tabous sont utilisées, la première est une liste tabou régulière qui empêche de revenir aux solutions déjà visitées, la deuxième contient les vecteurs poids. Hansen (1998) a proposé une recherche tabou basée sur l'idée de la méthode "Pareto Simulated Annealing" (Czyzak and Jaszkiewicz, 1998). La méthode utilise une population de solutions explorant chacune d'elles différentes régions de l'ensemble Pareto. Le vecteur poids est utilisé dans une fonction scalaire. De plus, à chaque solution est associée une liste tabou. La dispersion des solutions est assurée par la modification de leurs vecteurs poids. Pour chaque solution, la modification du vecteur poids vise à la déplacer des autres solutions proches dans l'espace des objectifs. Hansen a appliqué cette méthode au problème du Sac à Dos multi-objectif. Ben Abdelaziz and Krichen (1997) ont proposé un algorithme de recherche taboue multi-objectif conçu spécialement au problème du sac à dos multi-objectif. La méthode est guidée par des fonctions scalaires pondérées linéaires et par une relation de dominance. La relation de dominance est appliquée non seulement aux solutions mais aussi aux objets. La recherche locale est basée sur l'idée de l'insertion des objets non-dominés au sac à dos.

Algorithmes Évolutionnaires :

La première mise en œuvre d'un algorithme évolutionnaire multiobjectif (MOEA) remonte au milieu des années 1980. Depuis, une quantité considérable de recherches ont été faites dans ce domaine. L'importance croissante de ce domaine se traduit par une augmentation signicative, au cours des dix dernières années, de documents techniques à des conférences internationales et des revues, des livres, des séances spéciales à des conférences internationales et des groupes d'intérêt sur Internet (Coello, 2002).

La motivation principale de l'utilisation des EA pour résoudre des problèmes d'optimisation multiobjectif est que les EA traitent simultanément avec un ensemble de solutions possibles (dite population) qui nous permet de trouver

plusieurs membres de l'ensemble Pareto-optimale en une seule exécution de l'algorithme, au lieu d'effectuer une série d'essais séparés comme dans le cas des techniques mathématiques classiques.

4.3 Problématique & Approche de résolution

4.3.1 Enoncé du problème

Dans cette partie, nous considérons le problème de multicoloration de graphes modélisant le problème d'ordonnancement de jobs défini par sous les mêmes hypothèses et contraintes du problème décrit dans section 2.2, mais nous nous intéressons cette fois-ci à optimiser deux objectifs distincts, à savoir :

- Maximisation du gain total des sommets complétement colorés, définie par :

$$max(f1) = \sum_{i \in \mathrm{V}} (z_i.g_i)$$

$z_i = 1$ si le sommet i est complétement coloré, 0 sinon.

- Minimisation de la somme des plus grandes couleurs affectées aux sommets complètement colorés, définie par :

$$min(f2) = \sum_{i \in \mathrm{E}} \mathrm{M}ax_i$$

La décision est de trouver la meilleure affectation des n tâches de production aux machines avec l'objectif de maximiser le gain total des tâches exécutées ($\sum_{i \in \mathrm{V}} (z_i.g_i)$) et minimiser la somme des dates d'achèvement des tâches ($\sum_{i \in \mathrm{E}} \mathrm{C}_i$), sous les mêmes contraintes et hypothèses définies dans section 2.2. Le problème est NP-Difficile, puisqu'il s'agit d'une extension du problème de k-multi-coloration, qui est NP-difficile selon Malaguti and Toth (2010). Néanmoins, à notre connaissance, peu de travaux ont étudié le cas de deux

objectifs considérés conjointement pour un problème de multi-coloration. En effet, la plupart des travaux de la littérature considèrent soit le makespan (la date d'achèvement qui représente directement la productivité), soit d'autres objectifs comme la minimisation de la somme des retards.

La tendance des travaux de recherche pour la résolution de problèmes similaires est orientée vers les méthodes approchées. Le choix de ce type de méthodes est motivé par leur simplicité d'implémentation, et généralement par leur faible complexité temporelle et une bonne qualité de leurs solutions fournies. Parmi les travaux qui ont surmonté les difficultés liées à l'optimisation des problèmes d'ordonnancement multi-objectif. Nous citons, entre autres, les travaux de Gupta et al. (2001), où les auteurs ont procédé à la résolution du problème d'ordonnancement à deux machines en optimisant la somme des dates de fin d'exécution tout en ayant une plus petite valeur du makespan. Leurs heuristiques sont développées et évaluées empiriquement quant à leur efficacité à trouver un ordonnancement optimal pour le problème. T'Kinkt et al. (2002) ont abordé la même problématique à l'aide d'un algorithme de recherche locale qui combine une méthode exacte et une heuristique.

Concernant les problèmes de coloration de graphes, il existe de nombreux travaux dans lesquels le problème de coloration de graphes ou ses extensions sont utilisés pour modéliser des problèmes d'ordonnancement. Ceci est particulièrement pertinent lorsque les tâches sont incompatibles. Quelques exemples récents peuvent être trouvés dans Meuwly et al. (2010), Giaro et al. (2009) et Epstein et al. (2009). Parmi les applications de la multicoloration aux problèmes d'ordonnancement multi-objectif, Huang et al. (2017) sont intéressés à l'étude des systèmes en réseau homogènes. Ces derniers sont exposés à un risque élevé de subir des attaques malveillantes. Les auteurs présentent une approche pour améliorer la capacité de fonctionnement des systèmes en réseau via la diversité des logiciels en proposant un algorithme basé sur la multi coloration multiobjectif qui attribue stratégiquement divers progiciels aux hôtes du réseau. Ils ont montré qu'en attribuant à chaque machine un certain nombre de logiciels appropriés, ils peuvent efficacement isoler les

hôtes infectés, ce qui disperse l'infection réelle par le ver. Comme l'attaque de type ver ne peut se propager d'un nœud à l'autre qu'en suivant une arête défectueuse, ils ont modélisé la tâche d'affectation de logicielle par un problème de multicoloration de graphes en attribuant un ensemble de couleurs à chaque nœud pour éviter que les nœuds adjacents ne partagent la même couleur tout en respectant les contraintes réelles du problème.

Dans ce qui suit, nous allons proposer une approche basée sur la recherche tabou, suivie d'une adaptation de l'approche NSGA-II (Non-dominated Sorting Genetic Algorithm II) à la résolution du problème.

4.3.2 Approche de résolution : Tabu bi-objectif

4.3.2.1 Principe général

La recherche tabou (TS : Tabu Search) est une méta-heuristique qui a été appliquée avec succès à la résolution de problèmes combinatoires. Cette approche est proposée par (Glover, 1986). Son idée de base consiste à introduire la notion de mémoire dans la stratégie d'exploration de l'espace de solutions réalisables. C'est une procédure itérative qui, partant d'une solution initiale (cette solution peut être construite par une heuristique ou générée aléatoirement), tente de converger vers la meilleure solution en exécutant à chaque itération un mouvement dans l'espace de recherche. Chaque itération consiste d'abord à explorer un ensemble de solutions voisines d'une solution courante pour ensuite en choisir la meilleure (Algorithme-4.5).

La notion de mémoire consiste à utiliser une mémoire afin de conserver pendant un moment les informations sur les solutions déjà visitées. Ces informations sont déclarées taboues et elles sont stockées dans une liste de longueur donnée, appelée la «***liste des tabou***» ou la «***mémoire tabou***» (d'où le nom à la méthode). Les informations données par la liste tabou sont utilisées pour établir une restriction appelée «restriction tabou», qui permet de classer certains mouvements comme étant interdits et permet ainsi d'éviter de retourner à des solutions déjà visitées dans un passé récent, ainsi le phénomène de cy-

Algorithme 4.5 Algorithme Tabu Search

```
1: Choisir une solution initiale x ∈ Ω;
2: Mettre x* = x;
3: Générer N(x) voisinage de la solution x;
4: Mettre T = ∅ (T : Liste Tabou);
5: Pour t = 0 : |N(x)|, faire
6:    Choisir la meilleure solution y ∈ N(x) tel que y ∉ T;
7:    Mettre x = y;
8:    T = T ∪ x (Mettre à jour la liste tabou);
9:    Si f(x) < f(x*)
10:      x* = x;
11:   Finsi
12:   Retourner la meilleure solution x*;
13: Finpour
```

clage est évité.

La procédure s'arrête lorsqu'une condition d'arrêt est satisfaite, généralement après un nombre fixé d'itérations ou encore après un nombre d'itérations sans amélioration de la solution. Pour certains problèmes d'optimisation, la recherche tabou donne de bons résultats. De plus, dans sa forme de base, la recherche tabou contient moins de paramètres que les autre métaheuristiques, ce qui la rend simple à utiliser.

4.3.2.2 Algorithme Tabu bi-objectif pour le problème étudié

Dans cette section, nous présentons une approche basée sur la recherche tabou bi-objectif pour explorer l'espace des solutions réalisables via plusieurs directions, où à chaque itération un ensemble de taille fixe de solutions courantes est manipulé. Chacune de ces solutions possède sa propre liste tabou qui doit être aussi de taille fixe. C'est une stratégie qui permet de lancer une recherche de solutions non-dominées dans plusieurs directions afin d'explorer au maximum l'espace des solutions réalisables du problème traité.

En effet, l'algorithme part d'un ensemble de solutions non dominées S, tel que $|S| = D_{max}$ est le nombre de directions à explorer. Pour chaque solution $x_k \in S$, le voisinage $N(x_k)$ est construit et l'ensemble de solutions non-dominées ENSD est mis à jour. Ensuite, les solutions non taboues sont sélec-

tionnées $S' = \cup_{k=1}^{D_{max}} N(x_k)$ pour qu'elle soient explorer à la prochaine itération et les solutions restantes sont stockées dans une liste de candidats CL en tant que solutions potentielles à explorer.

Les étapes de l'approche Tabu bi-objectif sont données dans l'Algorithme 4.6.

Algorithme 4.6 Algorithme Tabu Search bi-objectif **BITS**

Entrées : S tel que $|S| = D_{max}$, Taille du voisinage, Critère d'arrêt
Sorties : ESND (ensemble des solutions non-dominées)

1. Initialisation
   $ESND = S$; $CL = \emptyset$; $i = 0$.
   $TL_{x_k} = \emptyset$, $\forall k = 1, ..., N_{max}$ (TL_{x_k} : Liste taboue associée à la solution x_k)
2. $S' = \emptyset$
3. **Pour** $x_k^i \in S$ **faire**
4. **Pour** chaque solution voisine $y \in N(x_k^i)$ **faire**
   Si $(y \notin TLx_k^i)$
   ESND = Solutions non-dominées de $(ESND \cup y)$
   S' = Solutions non-dominées de $(S' \cup y)$
5. $CL = \emptyset$
6. **Si** $(|S'| \leq D_{max})$ ***Aller à l'étape 7***
   Sinon
   Choisir D_{max} solutions depuis l'ensemble S'
   Enregistrer les solutions non-séléctionées dans la liste CL
7. **Pour** chaque solution $y \in S'$ **faire**
   $TL_x = TL_x \cup y$ (y est voisine de x_k)
8. $S = S'$
9. **Si** critère d'arrêt non satisfait
   $i = i + 1$ ***Aller à l'étape 2***

Fin Algorithme

Le codage des solutions est l'un des facteurs les plus importants qui affectent les performances de n'importe quel algorithme. Ce dernier doit garantir la possibilité d'explorer entièrement l'espace des solutions réalisables du problème lors de la création de nouvelles solutions. Deux types de représentations sont distingués : le codage direct et le codage indirect. Dans un codage direct, la représentation d'une solution est complète du problème traité. Ainsi, chaque représentation contient toutes les informations utiles à la création de la solution. Dans le codage indirect, les solutions ne sont pas représentées directement mais plutôt par des listes de priorités ou des heuristiques.

Dans ce travail, nous avons utilisé un codage indirect basé sur un vecteur d'entier de taille n, où n est le nombre de sommets du graphe des conflits. Ce vecteur considère l'ordre de traitement (coloration) des sommets lors de la reconstruction des solutions (Figure 4.2). La construction d'une solution et son évaluation se fait par la coloration des sommets du graphe de conflits du problème considéré un par un en suivant l'ordre du vecteur de priorité, par l'attribution à chaque sommet les plus petites couleurs non encore attribuées à ses voisins (voir chapitre 2).

5	4	6	8	10	7	3	1	2

FIGURE 4.2 – Codage par un vecteur entier

Cette représentation permet de créer pour chaque solution x un voisinage $\mathrm{N}(x) \subset \mathrm{X}$, et chaque solution $x' \in \mathrm{N}(x)$ est atteinte à partir de x par une opération élémentaire appelée mouvement.

Parmi les mouvements possibles susceptibles d'engendrer des solutions voisines réalisables, nous citons :

- l'inversion de deux éléments successifs dans la solution courante.
- la permutation de deux éléments distincts.
- le déplacement d'un élément de sa place d'origine à une nouvelle place.

Algorithme 4.7 Génération de voisinage

Entrées :

- Une solution s
- Taille du voisinage
- Stratégie

Sorties : $\mathrm{N}(s)$ voisinage de s

tantque le nombre de solutions visitées est inférieur à la taille de voisinage **faire**
Générer une solution non taboue s' en appliquant la stratégie dans s
$\mathrm{N}(s) \leftarrow \mathrm{N}(s) \cup \mathrm{N}(s')$
fin tantque

Pour la génération du voisinage d'une solution courante au niveau de l'algorithme BITS, nous avons adopté une structure de mémoire à court terme et

utilisé ensuite deux types de mouvements : (i) la permutation de deux éléments distincts et (ii) le déplacement d'un élément de sa position d'origine vers une nouvelle position. Ces mouvements s'alternent entre eux selon la stratégie développée dans l'Algorithme 4.7.

4.3.3 Approche de résolution : NSGA-II

4.3.3.1 Principe général

L'algorithme NSGA-II (Non-dominated Sorting Genetic Algorithm II) est l'un des algorithmes évolutionnaires les plus compétitifs en optimisation multi-objectif il a été proposé par Deb (2001), depuis il a été utilisé dans divers domaines en pratique. Il s'agit d'un algorithme évolutionnaire élitiste de type Pareto qui calcule une approximation de l'ensemble des solutions non dominées appelé aussi front Pareto ou surface de compromis, en se basant sur le concept de non dominance. Au départ, l'algorithme génère aléatoirement une population initiale P_0 de taille N. À chaque génération, une sélection par ***tournoi binaire*** est effectuée afin de générer, à partir d'une population P_t, N nouveaux individus. La sélection par tournoi binaire est un mécanisme qui permet de sélectionner deux individus au hasard dans la population, leurs objectifs sont comparés, et le meilleur d'entre eux est sélectionné. Ce processus est répété afin d'obtenir autant d'individus que nécessaire. De nouveaux individus sont obtenus grâce à la stratégie de reproduction (croisements/mutations) appliquée sur les N individus sélectionnés avec un pourcentage de chance déterminé, ainsi une population des descendants Q_t est créée.

Par la suite, les deux populations sont combinées pour former une nouvelle population R_t de taille 2N. Ensuite, la population R_t est triée en différents fronts non-dominés en utilisant le principe de dominance. La nouvelle population est remplie par des solutions de différents fronts non-dominées Figure 4.3. Le remplissage commence par le premier front non-dominé F_1 et se poursuit avec les points de second front non-dominé. Chaque front est ainsi sélectionné (par ordre croissant) jusqu'à ce qu'un front ne puisse pas

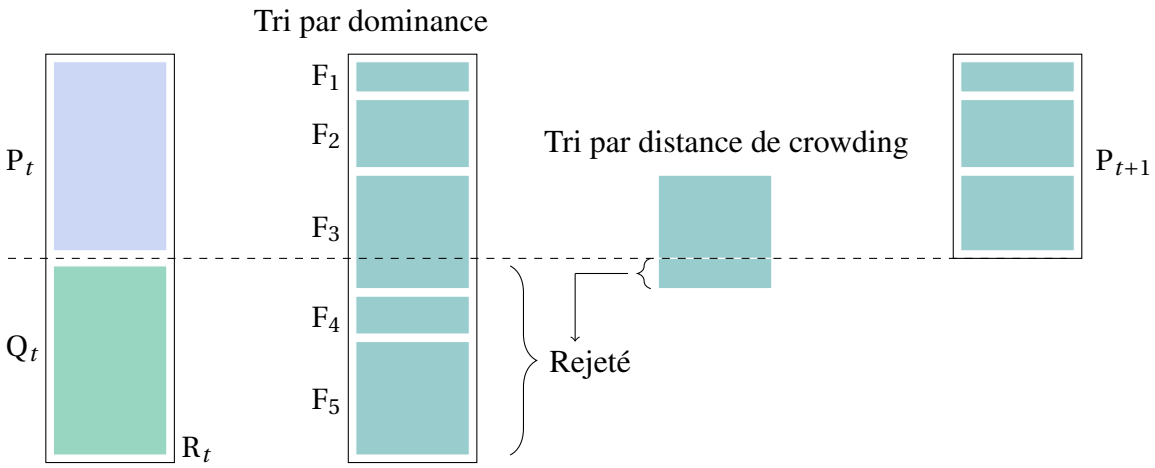

FIGURE 4.3 – Principe général de l'algorithme NSGA-II

intégrer entièrement la population ($|F_i| > |P| - \sum_{j=0}^{i-1} |F_j|$ avec F_i le front considéré et P la population). Ce front est alors trié selon la ***crowding distance*** et seuls les meilleurs individus en fonction de ce score sont sélectionnés (Deb et al., 2002). Le tri par la crowding distance a pour objectif de préserver la di-

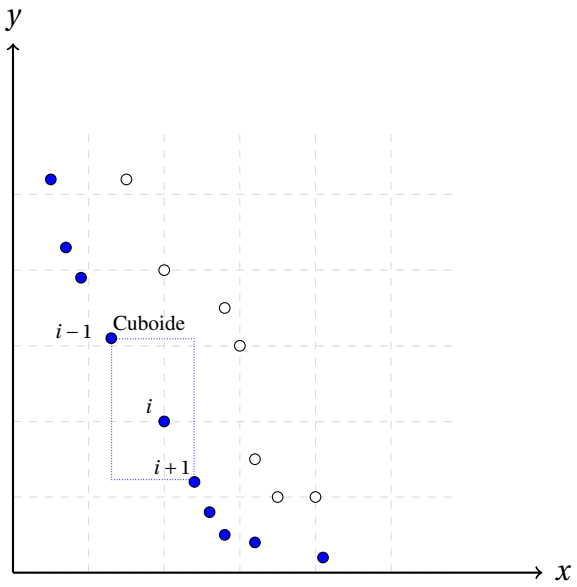

FIGURE 4.4 – Calcul de la distance Crowding

versité dans la population. Pour cela, il calcule pour chaque individu un rang en fonction de la proximité de son score par rapport aux individus ayant les scores les plus proches, plus un individu est éloigné des autres, meilleur est son classement. Cette distance est calculée en fonction des scores des individus pour les fonctions objectifs (Figure 4.4). Pour chaque fonction objectif, les chromosomes sont triés et leurs scores de distances sont calculés à partir

de la distance euclidienne qui les sépare de leurs voisins les plus proches. Les extremums sont tous conservés, et la distance finale d'un individu est la somme de ses distances pour chaque fonction objectif. Plusieurs générations sont ainsi créées, jusqu'à obtention d'une population finale (cf. algorithme 4.8).

Algorithme 4.8 Algorithme de NSGA-II

1: $t \leftarrow 0$
2: Générer une population initiale P(t) de solutions aléatoires
3: **Tant que** le critère n'est pas satifait **faire**
4: Sélectionner les parents pour faire les opérations génétiques
5: Produire Q(t) en appliquant les opérateurs génétiques
6: R(t) $\leftarrow$ P(t) $\cup$ Q(t)
7: Classer R(t) en différents fronts non-dominé
8: Tri par dominance et par la distance Crowding des N meilleurs individus de R(t)
9: Mettre les meilleurs individus dans P(t+1)
10: $t \leftarrow t+1$
11: **fin tant que**

4.3.3.2 Composantes de l'algorithme NSGA-II

Les opérateurs jouent un rôle prépondérant dans la possible réussite d'un algorithme génétique. Nous en dénombrons trois principaux : l'opérateur de sélection, de croisement et de mutation. Si le principe de chacun de ces opérateurs est facilement compréhensible, il est toutefois difficile d'expliquer l'importance isolée de chacun de ces opérateurs dans la réussite de l'algorithme génétique. A l'instar de ces opérateurs, la première tâche lors de l'implémentation d'un algorithme génétique consiste à trouver une représentation adéquate des individus représentant l'espace des solutions du problème abordé.

- **Codage et évaluation des solutions :**
  Dans ce travail nous avons utilisé le même codage utilisé dans la section précédente, ce codage basé sur un vecteur entier de taille n, où n est le nombre de sommets de graphe des conflits. Ce vecteur considère l'ordre de traitement (coloration) des sommets lors de la reconstruction des solutions (Figure 4.2).

- **Stratégie de sélection :**
  Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. La sélection est un processus qui consiste à choisir parmi tous les individus de la population ceux qui vont participer à la construction d'une nouvelle génération, où ce choix est basé essentiellement sur les valeurs d'adaptation de chaque individu. Un individu ayant une grande valeur de la fonction objectif aura plus de chance d'être sélectionné pour participer à la reproduction de la prochaine génération.

  Dans ce travail nous avons doté notre algorithme avec une sélection par tournoi qui consiste à comparer une paire d'individus choisis au hasard, où le meilleur de ces deux est gagnant et sera déclaré sur le tournoi, avec une population de N chromosomes, on forme N paires de chromosomes pour la reproduction.
- **Opérateur de croisement :**
  La méthode de croisement à deux points est implémentée comme proposée dans Ishibuchi and Murata. (1998). Le premier (resp. le second) enfant est généré en conservant les extrémités de chromosome du premier parent et en complétant avec les cases manquantes suivant leur ordre d'apparition dans le deuxième (resp. le premier) parent, tel qu'il est illustré sur figure 4.5.

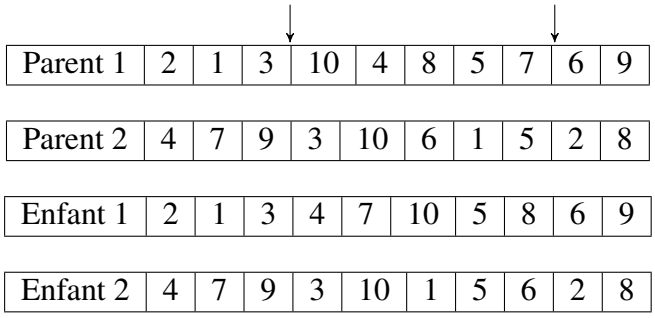

FIGURE 4.5 – Croisement à deux points

- **Opérateur de mutation** La méthode de mutation par échange est implémentée comme proposée par Black et al. (2000). Elle consiste à sélectionner de manière aléatoire deux gènes d'un individu et à échanger

les positions respectives des deux éléments choisis, tel qu'il est illustré sur figure 4.6.

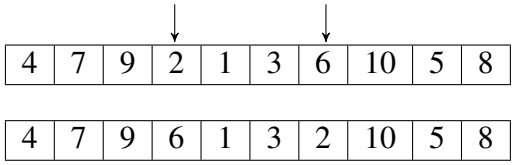

FIGURE 4.6 – Mutation par échange

4.4 Expérimentations et résultats

Dans cette section, nous donnons un aperçu rapide sur l'implémentation des approches proposées pour la résolution du problème de multicoloration de graphes modélisant l'ordonnancement multi-objectif. Nous précisons en premier lieu les pramètres de la recherche Tabu. Puis, fixons les differents paramètres des opérateurs de la méthode NSGA-II. Pour les instances nous avons utilisé les benchmarks proposés dans le travaux de Thevenin et al. (2018b).

4.4.1 Implémentation et paramétrage des approches

4.4.1.1 Approche 1 : Tabu bi-objectif

Comme nous l'avons précisé dans la sous section 4.3.2.1, l'approche Tabou search contient moins de paramètres à fixer que la plupart des métaheuristiques de la littérature. Ces paramètres jouent un rôle extrêmement important pour renforcer son efficacité. Un paramétrage pertinent, tel que la taille de l'ensemble des solutions manipulé à chaque itération impacte la réussite de l'algorithme. Si elle est très grande l'algorithme permet une meilleure exploration de l'espace des solutions réalisables, néanmoins le temps de calcul devient très important. Donc un choix intermédiaire s'impose dans une telle situation.

Pour ce faire, et afin de mieux paramétrer l'approche **BITS**, des expérimen-

tations préliminaires ont était effectuées pour arriver à fixer le nombre de solutions explorées à chaque itération D_{max} à 50. Egalement, pour chacune de ces solutions, la taille du voisinage est fixée 100 à (N = 100), et finalement la taille de la liste tabou a été maintenue à 40 (TL_x = 40).

4.4.1.2 Approche 2 : NSGA-II

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres fixés à l'avance. Un choix judicieux de ces paramètres, tels que la taille de la population, les taux de croisement et de mutation ainsi que le nombre de générations permet d'obtenir des solutions de qualité. Par exemple si la taille de la population est trop grande le temps de calcul de l'algorithme peut s'avérer très important, et si elle est trop petite, il peut converger trop rapidement vers de mauvais fronts Pareto.

Généralement il n'existe pas de valeurs typiques acceptées universellement pour tous ces paramètres. Mais la majorité des études utilisant les algorithmes génétiques montrent une certaine tendance pour certaines valeurs sur la base d'expérimentations. Par exemple, le taux de croisement est souvent choisi à 0.8 et le taux de mutation est généralement choisi à 0.3 comme utilisé dans les travaux de Deb (2001).

Pour nos expérimentations, nous fixons le taux de croisement à 0.8. Le taux de mutation est fixé à 0.3, concernant la taille de la population nous l'avons fixé à 200 individus. L'algorithme **NSGA-II** pour résoudre le problème abordé est implémenté en C++ et est inspirée de la bibliothèque en ligne de l'université du Michigan (États-Unis) [1].

4.4.2 Comparison & résultats

Dans le but d'évaluer la performance de l'approche Tabu proposée sur le problème étudié, nous l'avons comparé à l'approche **NSGA-II** sur deux métriques de performance, à savoir, (C) : L'ensemble couvrant (Zitzler, 1999)

1. https ://www.egr.msu.edu/ kdeb/codes.shtml

et (R) : le ratio des solutions non dominées dans l'ensemble de solutions approchées non dominées globales (Altiparmak et al., 1999).

La première métrique mesure la qualité de convergence d'un ensemble de solutions. Pour sa définition, soit S_1 et S_2 deux ensembles de solutions non dominées. $C(S_1, S_2)$, dans l'équation 4.5, donne le pourcentage de solutions dans S_2 qui sont dominées par au moins une solution dans S_1.

$$C(S_1, S_2) = \frac{|y \in S_2| \ \exists x \in S_1 : x \succ y}{|S_2|} \tag{4.5}$$

Où le symbole $\succ$ désigne l'opérateur de domination. Cela signifie pour $x \succ y$ que la solution x domine la solution y. Ainsi, $C(S_1, S_2) = 1$ implique que toutes les solutions dans S_2 sont dominées par au moins une solution dans S_1 tandis que $C(S_1, S_2) = 0$ montre qu'aucune solution dans S_2 n'est dominée par une solution dans S_1. Cependant, $C(S_1, S_2) > C(S_2, S_1)$ indique que l'ensemble S_1 est meilleur que S_2. Il est a noté $C(S_1, S_2) \neq 1 - C(S_2, S_1)$.

La deuxième métrique, donnée dans l'équation 4.6, calcule le rapport des solutions dans $S_i, i \in \{1, ..., 2\}$ qui ne sont dominés par aucune autre solution dans S, où S est l'union de S_i qui ne comprend que des solutions non dominées. Cette métrique explique que plus le rapport $R(S_i)$ est élevé, meilleur est l'ensemble de solutions S_i.

$$R(S_i) = \frac{|S_i - \{x \in S_i | \exists y \in S : x \succ y\}|}{|S_i|} \tag{4.6}$$

Les deux approches, nommées S_1 pour NSGA-II et S_2 pour BITS, sont implémentées en C++ et sont exécutées sur un processeur Intel Quad-core i7 avec 8 Go de mémoire RAM DDR3. Le temps CPU maximum autorisé (en seconde) est de 60.n (n est un nombre de sommets), Les résultats expérimentaux concernant ces deux mesures de performance utilisées (C-métrique et R-métrique) sont présentés dans les tableaux 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 et 4.8.

Les tableaux 4.1, 4.2, 4.3, 4.4, 4.5 et 4.6 décrivent les résultats des instances de tailles 10, 25, 50, 100, 300 et 500 respectivement. La colonne 1 fait réfé-

rence au nom de l'instance et à sa taille correspondante. Les colonnes 2, 3, 4 et 5 rapportent le nombre de solutions non-dominées initialement retournées par l'approche NSGA-II (resp. BITS)Les colonnes de 6 à 9 correspondent aux résultats des métriques de comparaison utilisées. Les tableaux 4.7 et 4.8 résument les moyennes des résultats obtenus pour les deux métriques pour chaque instance du problème.

En analysant les résultats du tableau 4.7 relatif aux petites instances, on constate $C(S1, S2) = 0.15$, donc l'ensemble des solution non-dominées de l'approche NSGA-II domine ou élimine 15% des solution non-dominées de l'approche BITS, alors que $C(S2, S1) = 0.18$, et donc l'ensemble des solution non-dominées de l'approche BITS domine ou élimine 18% des solution non-dominées de l'approche NSGA-II. Concernant la deuxième métrique R nous constatons une égalité des nombres de solutions non-dominées des deux approches proposées.

En ce qui concerne les grandes instances sur le tableau 4.8, $C(S1, S2) = 0.16$, donc l'ensemble des solution non-dominées de l'approche NSGA-II domine ou élimine 16% des solution non-dominées de l'approche BITS, alors que $C(S2, S1) = 0.50$, et donc l'ensemble des solution non-dominées de l'approche BITS domine ou élimine 50% des solutions non-dominées de l'approche NSGA-II. Nous remarquons alors que l'approche BITS est plus performante en terme de qualité des solution non-dominées obtenues. D'autre part, pour la seconde métrique R nous constatons que l'approche BITS dépasse l'approche NSGA-II avec un taux de 58% des solutions non-dominées et juste 42% pour NSGA-II.

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
10_0,2_a	1	1	1	1	0,00	0,00	0,50	0,50
10_0,2_b	3	3	3	3	0,00	0,00	0,50	0,50
10_0,2_c	1	1	1	1	0,00	0,00	0,50	0,50
10_0,2_d	2	2	2	2	0,00	0,00	0,50	0,50
10_0,2_e	5	5	5	5	0,00	0,00	0,50	0,50
10_0,5_a	3	3	3	3	0,00	0,00	0,50	0,50
10_0,5_b	6	6	8	8	0,00	0,00	0,43	**0,57**
10_0,5_c	4	4	3	3	0,00	0,00	**0,57**	0,43
10_0,5_d	3	3	1	1	0,00	0,00	**0,75**	0,25
10_0,5_e	3	3	3	1	**0,67**	0,00	**0,75**	0,25
10_0,8_a	5	5	5	5	0,00	0,00	0,50	0,50
10_0,8_b	8	8	6	6	0,00	0,00	**0,57**	0,43
10_0,8_c	5	5	5	5	0,00	0,00	0,50	0,50
10_0,8_d	4	4	4	4	0,00	0,00	0,50	0,50
10_0,8_e	4	4	4	4	0,00	0,00	0,50	0,50

TABLEAU 4.1 – Résultats des instances bi-objectif de 10 sommets

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
25_0,2_a	16	14	17	17	0,00	**0,13**	0,45	**0,55**
25_0,2_b	14	12	15	14	0,07	**0,14**	0,46	**0,54**
25_0,2_c	17	17	18	18	0,00	0,00	0,49	**0,51**
25_0,2_d	19	17	18	18	0,00	**0,11**	0,49	**0,51**
25_0,2_e	23	23	24	24	0,00	0,00	0,49	**0,51**
25_0,5_a	22	21	25	25	0,00	**0,05**	0,46	**0,54**
25_0,5_b	27	26	27	21	**0,22**	0,04	**0,55**	0,45
25_0,5_c	33	30	32	23	**0,28**	0,09	**0,57**	0,43
25_0,5_d	19	19	22	19	**0,14**	0,00	0,50	0,50
25_0,5_e	26	16	38	21	**0,45**	0,38	0,43	**0,57**
25_0,8_a	16	15	17	13	**0,24**	0,06	**0,54**	0,46
25_0,8_b	27	24	20	17	**0,15**	0,11	**0,59**	0,41
25_0,8_c	15	11	11	10	0,09	**0,27**	**0,52**	0,48
25_0,8_d	33	23	29	26	0,10	**0,30**	0,47	**0,53**
25_0,8_e	23	19	34	28	**0,18**	0,17	0,40	**0,60**

TABLEAU 4.2 – Résultats des instances bi-objectif de 25 sommets

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
50_0,2_a	62	54	76	19	**0,75**	0,13	**0,74**	0,26
50_0,2_b	54	19	60	53	0,12	**0,65**	0,26	**0,74**
50_0,2_c	67	55	66	50	**0,24**	0,18	**0,52**	0,48
50_0,2_d	58	50	61	46	**0,25**	0,14	**0,52**	0,48
50_0,2_e	46	46	28	24	**0,14**	0,00	**0,66**	0,34
50_0,5_a	50	14	50	40	0,20	**0,72**	0,26	**0,74**
50_0,5_b	36	10	38	38	0,00	**0,72**	0,21	**0,79**
50_0,5_c	46	19	42	32	0,24	**0,59**	0,37	**0,63**
50_0,5_d	61	47	66	20	**0,70**	0,23	**0,70**	0,30
50_0,5_e	56	35	56	45	0,20	**0,38**	0,44	**0,56**
50_0,8_a	46	34	33	19	**0,42**	0,26	**0,64**	0,36
50_0,8_b	59	21	39	39	0,00	**0,64**	0,35	**0,65**
50_0,8_c	45	34	65	27	**0,58**	0,24	**0,56**	0,44
50_0,8_d	55	33	40	26	0,35	**0,40**	**0,56**	0,44
50_0,8_e	40	10	37	35	0,05	**0,75**	0,22	**0,78**

TABLEAU 4.3 – Résultats des instances bi-objectif de 50 sommets

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
100_0,2_a	200	74	105	84	0,20	**0,63**	0,47	**0,53**
100_0,2_b	200	61	141	110	0,22	**0,70**	0,36	**0,64**
100_0,2_c	196	92	74	37	0,50	**0,53**	**0,71**	0,29
100_0,2_d	200	65	111	63	0,43	0**,68**	**0,51**	0,49
100_0,2_e	200	46	117	105	0,10	**0,77**	0,30	**0,70**
100_0,5_a	200	37	54	48	0,11	**0,82**	0,44	**0,56**
100_0,5_b	200	166	61	31	**0,49**	0,17	**0,84**	0,16
100_0,5_c	200	137	47	28	**0,40**	0,32	**0,83**	0,17
100_0,5_d	199	68	78	54	0,31	**0,66**	**0,56**	0,44
100_0,5_e	198	25	77	68	0,12	**0,87**	0,27	**0,73**
100_0,8_a	200	33	57	50	0,12	**0,84**	0,40	**0,60**
100_0,8_b	200	37	66	65	0,02	**0,8**2	0,36	**0,64**
100_0,8_c	199	115	98	44	**0,55**	0,42	**0,72**	0,28
100_0,8_d	197	7	62	62	0,00	**0,96**	0,10	**0,90**
100_0,8_e	196	55	66	53	0,20	**0,72**	**0,51**	0,49

TABLEAU 4.4 – Résultats des instances bi-objectif de 100 sommets

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
300_0,2_a	74	47	42	42	0,00	**0,36**	**0,53**	0,47
300_0,2_b	81	38	69	69	0,00	**0,53**	0,36	**0,64**
300_0,2_c	72	56	56	46	0,18	**0,22**	**0,55**	0,45
300_0,2_d	74	57	84	62	**0,26**	0,23	0,48	**0,52**
300_0,2_e	74	58	74	61	0,18	0,22	**0,49**	**0,51**
300_0,5_a	71	6	48	48	0	0,92	0,11	0**,89**
300_0,5_b	66	17	47	47	0	0,74	0,27	**0,73**
300_0,5_c	79	14	68	68	0	0,82	0,17	**0,83**
300_0,5_d	71	37	72	72	0	0,48	0,34	**0,66**
300_0,5_e	68	64	60	60	0	0,06	**0,52**	0,48
300_0,8_a	44	11	33	33	0,00	**0,75**	0,25	**0,75**
300_0,8_b	37	16	28	28	0,00	**0,57**	0,36	**0,64**
300_0,8_c	37	12	44	44	0,00	**0,68**	0,21	**0,79**
300_0,8_d	47	21	58	44	0,24	**0,55**	0,32	**0,68**
300_0,8_e	38	24	49	48	0,02	**0,37**	0,33	**0,67**

TABLEAU 4.5 – Résultats des instances bi-objectif de 300 sommets

	NSGA-II		TABU		Critères de comaprison			
Instances	Sol_{ini}	Sol_{fin}	Sol_{ini}	Sol_{fin}	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
500_0,2_a	51	16	30	30	0,00	**0,69**	0,35	**0,65**
500_0,2_b	73	73	59	28	**0,53**	0,00	**0,72**	0,28
500_0,2_c	57	46	59	52	0,12	**0,19**	0,47	**0,53**
500_0,2_d	62	54	61	54	0,11	**0,13**	0,50	0,50
500_0,2_e	82	50	48	39	0,19	**0,39**	**0,56**	0,44
500_0,5_a	40	28	57	57	0,00	**0,30**	0,33	**0,67**
500_0,5_b	62	0	33	33	0,00	**1,00**	0,00	**1,00**
500_0,5_c	69	22	31	31	0,00	**0,68**	0,42	**0,58**
500_0,5_d	37	34	65	49	0,25	0,08	**0,41**	**0,59**
500_0,5_e	54	54	35	31	**0,11**	0,00	**0,64**	0,36
500_0,8_a	64	5	18	18	0,00	**0,92**	0,22	**0,78**
500_0,8_b	30	28	83	51	**0,39**	0,07	0,35	**0,65**
500_0,8_c	40	32	73	50	**0,32**	0,20	0,39	**0,61**
500_0,8_d	38	23	62	51	0,18	**0,39**	0,31	**0,69**
500_0,8_e	35	33	53	29	**0,45**	0,06	**0,53**	0,47

TABLEAU 4.6 – Résultats des instances bi-objectif de 500 sommets

		Critères de comaprison			
Instances	d	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
10	0,2	0,00	0,00	0,50	0,50
	0,5	0,13	0,00	0,60	0,40
	0,8	0,00	0,00	0,51	0,49
	Average	0,04	0,00	0,54	0,46
25	0,2	0,01	0,07	0,47	0,53
	0,5	0,22	0,11	0,50	0,50
	0,8	0,15	0,18	0,50	0,50
	Average	0,13	0,12	0,49	0,51
50	0,2	0,30	0,22	0,54	0,46
	0,5	0,27	0,53	0,40	0,60
	0,8	0,28	0,46	0,47	0,53
	Average	0,28	0,40	0,47	0,53
Average		0,15	0,18	0,50	0,50

TABLEAU 4.7 – Récapitulatif des résultats de petites instances

		Critères de comaprison			
Instances	d	C(S1, S2)	C(S2, S1)	R(S1)	R(S2)
100	0,20	0,29	0,66	0,47	0,53
	0,50	0,29	0,57	0,59	0,41
	0,80	0,18	0,75	0,42	0,58
	Average	0,25	0,66	0,49	0,51
300	0,20	0,12	0,31	0,48	0,52
	0,50	0,00	0,60	0,28	0,72
	0,80	0,05	0,58	0,30	0,70
	Average	0,06	0,50	0,35	0,65
500	0,20	0,19	0,28	0,52	0,48
	0,50	0,07	0,41	0,36	0,64
	0,80	0,27	0,33	0,36	0,64
	Average	0,18	0,34	0,41	0,59
Average		0,16	0,50	0,42	0,58

TABLEAU 4.8 – Récapitulatif des résultats de grandes instances

4.5 Conclusion

Nous avons, dans ce chapitre étudié le problème de multi-coloration de graphes modélisant le problème d'ordonnancement bi-objectif au sens de l'ensensemble des solutions non dominées. Les deux objectifs à optimiser simultanément sont la maximisation du gain total des jobs effectués et la minimisation de la somme des dates d'achèvement des jobs (total flow time ou total completion time). Nous avons présenté dans ce chapitre les principaux concepts de l'optimisation multiobjectif et quelques travaux de recherche sur l'optimisation de problèmes multiobjectifs (ordonnancement et multi-coloration). Ensuite, nous avons présenté les deux métaheuristiques (Recherche taboue bi-objectif **BITS** et **NSGA-II**) proposées pour la résolution du problème étudié. Des expérimentations sur des instances de la littérature ont été effectuées pour valider les deux approches proposées, en utilisant un indicateur basé sur deux métriques de performance. L'analyse approfondie des résultats obtenus a montré que la Recherche taboue bi-objectif **BITS** est plus efficace et domine empiriquement le NSGA-II sur la majorité des benchmarks.

Conclusion générale et perspectives

Les travaux de cette thèse s'intéressent aux problèmes d'ordonnancement d'ateliers de production via la multicoloration de graphes. L'objectif est de résoudre le problème d'ordonnancement de jobs conflictuels en passant par sa réduction en un problème de multicoloration de graphes.

Nous avons, tout d'abord, présenté les différentes composantes des problèmes d'ordonnancement, à savoir (tâches, ressources, contraintes et critères), leur différents types d'ateliers de production (job-shop, flow-shop et open-shop). Nous avons également présenté un bref aperçu de quelques notions élémentaires de la thèorie de complexité et les différentes méthodes d'optimisation les plus souvent utilisées pour leur résolution : méthodes exactes (programmation linéaire, programmation dynamique) ou méthodes approchées : (recuit simulé, recherche tabou, algorithmes génétiques).

Par la suite, notre intérêt s'est porté sur la coloration de graphes, cette branche importante de la théorie des graphe qui consiste à attribuer une couleur à chaque sommet du graphe de manière que deux sommets reliés par une arête soient de couleurs différentes. Ceci offre une large possibilité de modéliser de nombreux problèmes d'ordonnancement, notamment lorsque certaines tâches sont soumises à des exigences d'incompatibilité, elles sont alors assignées à des classes de couleurs différentes. Cependant, dans certains cas l'attribution d'une seule couleur à chacun des sommets du graphe ne permet pas de modéliser certains problèmes pratiques. Ce type de problème peut être modélisé par la multicoloration de graphes, qui est extension de la coloration de graphes qui attribue non seulement une couleur mais un ensemble de couleurs à chacun des sommets de graphes.

Après avoir donné un aperçu global permettant une compréhension concise des problèmes d'ordonnancement et de coloration de graphes, nous sommes passés à la description du problème d'ordonnancement auquel nous nous intéressons, en présentant ses différentes contraintes, ses hypothèses ainsi que ses objectifs considérés. Ensuite, une réduction de ce problème ordonnancement à un problème de multicoloration de graphes est présentée, ainsi qu'un résumé de quelques travaux de littérature dédiés à la résolution des problèmes d'ordonnancement par la multicoloration de graphes. Etant donné que la problématique étudiée considère plusieurs objectifs, nous l'avons abordé en deux étapes, où la première étape traite la version mono-objectif obtenue par l'agrégation des objectifs et l'autre étape traite la version bi-objectives au sens de l'ensemble des solutions non-dominées.

Pour la résolution du problème de multicoloration mono-objectif modélisant le problème d'ordonnancement, nous avons proposé une approche recuit simulé **ISA** utilisant un mécanisme de refroidissement modifié permettant de mieux guider la température en favorisant les paliers les plus prometteurs de l'espace de solution réalisables. De plus, nous avons adopté une technique de coloration des sommets basée sur la minimisation du gain perdu afin de renforcer l'efficacité de l'approche proposée. Finalement, et afin de mesurer les performances de l'approche **ISA**, des expérimentations sur des benchmarks ont été réalisées, et la méthode a montré une meilleure performance par rapport aux performances des méthodes de littérature.

Concernant le problème bi-objective, nous avons proposé une approche de recherche tabou bi-objectif **BITS**. Cette approche explore l'espace des solutions réalisables via plusieurs directions, où à chaque itération un ensemble de taille fixe de solutions courantes est manipulé. Chacune de ces solutions possède sa propre liste tabou qui est aussi de taille fixe. C'est une stratégie qui permet de lancer une recherche de solutions non-dominées dans plusieurs directions afin d'explorer au maximum l'espace des solutions réalisables du problème traité. Pour valider l'approche proposée, nous l'avons comparer avec une approche de résolution des problèmes d'optimisation multi-objectif

NSGA-II. Les résultats ont montrés que l'approche **BITS** est plus performante en terme de qualité des solution non-dominées obtenues.

Les principales contributions de cette thèse se résument dans les points suivants :

- Proposition d'une stratégie de coloration des sommets d'un graphe de conflit modélisant le problème d'ordonnancement étudié.
- Proposition d'une approche de recuit simulé basée sur un mécanisme de refroidissement modifié pour la résolution du problème de multi-coloration de graphes.
- Proposition d'une approche de recherche tabou bi-objectif pour la multi-coloration de graphes.

Les travaux réalisés dans cette thèse ouvrent des perspectives à d'autres études pouvant être menées dans les directions suivantes :

- Etude de la multi-coloration contiguë (i.e les couleurs affectées à un sommet forment un intervalle contigu), qui modélise des problèmes d'ordonnancement avec des tâches non preemptives.
- Proposition de méthodes exactes pour la multi-coloration de graphes multi-objectifs.

Bibliographie

Abou Kasm, O., Mohandes, B., Diabat, A., and El Khatib, S. (2019). Exam timetabling with allowable conflicts within a time window. Computers & Industrial Engineering, 127 :263–273. 44

Al-Anzi, F., Sotskov, Y., Allahverdi, A., and Andreev, G. (2006). Using mixed graph coloring to minimize total completion time in job shop scheduling. Applied mathematics and computation, 182(2) :1137–1148. 45

Altiparmak, F., Gen, M., Lin, L., and Paksoy, T. (1999). A genetic algorithm approach for multi-objective optimization of supply chain networks. Computers and Industrial Engineering., 51(1) :196–215. 95

Appel, L. and Haken, W. (1977). Every planar map is four colorable : Part 1, discharging. Illinois. J. Maths., 21 :429–490. 34

Baaziz, A., Ait Haddadène, H., Oulamara, A., and Kouider, A. (2023). Scheduling preemptive jobs on parallel machines with a conflict graph : A graph multi-coloring approach. International Journal of Mathematics in Operational Research (IJMOR)., 25(1) :47–67. 51, 53

Bellman, R. (1957). Dynamic programming. Princeton University Press. 18

Ben Abdelaziz, F. and Krichen, S. (1997). A tabu search heuristic for multiple objective knapsack problems. Ructor Research Report., pages 28–97. 82

Blacewicz, J., Ecker, K., Pesch, E., Schmidt, G., and Weglarz, J. (1996). Scheduling computer and manufacturing processes. Springer-Verlag, Berlin. 30

Black, T., Fogel, D., and Michalewicz., Z. (2000). Evolutionnary computation i : basic algorithm and operators. New York, Taylor Francis. 92

Blöchliger, I. and Zufferey, N. (2013). Multi-coloring and job-scheduling with assignment and incompatibility costs. Annals of Operations Research, 211(1) :83–101. 47

Bo, H., Han, P., Lu, B., Zhao, C., and Wang, X. (2021). Online monitoring and collaborative scheduling method for wheelset cyber-physical production system : A wheelset manufacturing system case study from a chinese high-speed train enterprise. Advanced Engineering Informatics, 47(1) :101–210. 45

Bogle, I., Slota, G., Boman, E., Devine, K., and Rajamanickam, S. (2022). Parallel graph coloring algorithms for distributed gpu environments. Parallel Computing, 110. 44

Coello, C. A. C. (1996). Empirical study of evolutionary techniques for multiobjective optimization in engineering design. phD. Thesis, Department of Computer Science, Tulane University New Orleans. 79

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms : A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering., 191(11-12) :1245–128. 82

Cook, S. (1971). The complexity of theorem proving procedures, proceedings of the third annual acm symposium on theory of computing. ACM, New York., 2570 :151–158. 17

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing a metaheuristic technique for multiple-objective combinatorial optimisation. Journal of Multi-Criteria Decision Analysis, 7 :34–47. 81, 82

Dantzig, G., Fulkerson, R., and Selmer (1954). Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America, 2(4) :393–410. 17

Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. John Wiley Sons, Chichester. 89, 94

Deb, K., Pratap, A., Agarwal, S., and Meyarivan., T. (2002). A fast and elitist multiobjective genetic algorithm : Nsga-ii. IEEE Transactions on Evolutionary Computation., 6(2) :182–197. 90

Epstein, L., Halld´orsson, M., Levin, A., and Shachnai, H. (2009). Weighted sum coloring in batch scheduling of conflicting jobs. Algorithmica., 55 :643–665. 84

Fogel, L., Owens, A., and Walsh, M. (1966). Artificial intelligence through simulated evolution. New York : John Wiley,, 11.

Gamache, M., Hertz, A., and Ouellet, J. (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. Computers Operations Research, 34(8) :2384–2395. 45

Garey, M. e. D. J. (1979). Computers and intractability. a guide to the theory of np-completeness. W.H. Freeman and Company, San Francisco., 2570 :P 3, 12, 14, 16, 57, 59, 69, 168. 17

Geoffrion, A. (1968). Proper effeciency and the theory of vector maximization. Journal of Mathematical Analysis and Applications, 22(3) :618–630. 79

Giaro, K., Kubale, M., and Obszarski, P. (2009). A graph coloring approach to scheduling of multiprocessor tasks on dedicated machines with availability constraints. Discrete Applied Mathematics, 157(17) :3625–3630. 45, 84

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers and Opérations Research, 5 :533–549. 20, 85

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, Massachusetts. 78

Graham, R., Lawler, E., Lenstra, J., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. Annals of discrete mathematics, 5 :287–326. 28

Grandibleux, X., Mezdaoui, N., and Fréville, A. (1997). A multiobjective tabu search procedure to solve combinatorial optimization problems. advances in multiple objective and goal programming. Lecture Notes in Economics and Mathematical Systems, 455 :291–300. 81

Gupta, J., Neppali, V., and Werner., F. (2001). Minimizing total flow time in a two-machine flowshop problem with minimum makespan. International Journal of Production Economics., 69 :323–338. 84

Haimes, Y. Y., Lasdon, L. S., and Wismer, D. A. (1971). On a bicriterion formulation of the problems of intergrated system identification and system optimization. IEEE Trans. On Systems, Man and Cybernetics, pages 296–297. 79

Hajek, B. (1988). Cooling schedules for optimal annealing. Mathematics of operations research, 13 :311-329. 53

Hansen, M. (1998). Metaheuristics for multiple objective combinatorial optimization. Ph.D. Thesis, Technical University of Denmark, Lyngby. 82

Huang, C., Zhu, S., Guan, Q., and He, Y. (2017). A software assignment algorithm for minimizing worm damage in networked systems. Journal of Information Security and Applications., 35 :55–67. 84

Ishibuchi, H. and Murata., T. (1998). A multi-objective genetic local search algorithm and its application to flow shop scheduling. IEEE Transactions on Systems, Man and Cybernetics., 28 :392–403. 92

Jovanović, P., Pavlović, N., Belošević, I., and Milinković, S. (2020). Graph coloring-based approach for railway station design analysis and capacity determination. European Journal of Operational Research, 287(1) :348–360. 44

Karp, R. M. (1972). Reducibility among combinatorial problems. in complexity of computer computations. Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y.. New York : Plenum., pages 85–103. 16, 46

Kessler, C., Litzinger, S., and Keller, J. (2021). Crown-scheduling of sets of parallelizable tasks for robustness and energy-elasticity on many-core systems with discrete dynamic voltage and frequency scaling. Journal of Systems Architecture, 115 :101–999. 46

Kheiri, A., Lewis, R., Thompson, J., and Harper, P. (2021). Constructing operating theatre schedules using partitioned graph colouring techniques. Health Systems, 10(4) :286–297. 45

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. American association for the advancement of science, 220(4598) :671–680. 21, 50

Kouider, A. (2019). Coloration de graphes et ses applications aux problèmes d'ordonnancement. Thèse de Doctorat à Université des Sciences et de la Technologie Houari Boumediène - Alger, page 22.

Kouider, A. and Ait Haddadène, H. (2021). A bi-objective branch-and-bound algorithm for the unit-time job shop scheduling : A mixed graph coloring approach. Computers & Operations Research, 132 :105–319. 44

Kouider, A., Ait Haddadène, H., Ourari, S., and Oulamara, A. (2015). Mixed integer linear programs and tabu search approach to solve mixed graph coloring for unit-time job shop scheduling. IEEE International Conference on Automation Science and Engineering (CASE), pages 1177–1181. 45

Kouider, A., Ait Haddadène, H., Ourari, S., and Oulamara, A. (2017). Mixed graph colouring for unit-time scheduling. International Journal of Production Research, 55(6) :1720–1729. 44, 45

Liedlof, M. (2007). lgorithmes exacts et exponentiels pour les problèmes np-difficiles : domination, variantes et généralisations. Thèse de Doctorat à l'université Paul Verlaine – Metz, page 13. 16

Mahapatra, R., Samanta, S., Allahviranloo, T., and Pal, M. (2019). Radio fuzzy graphs and assignment of frequency in radio stations. Computational and Applied Mathematics, 38(3) :1–20. 44

Malaguti, E. and Toth, P. (2010). A survey on vertex coloring problems. Intl. Trans. in Op. Res., 17 :1–34. 83

Maqrot, S. (2020). Méthodes d'optimisation combinatoire en programmation mathématique. application à la conception des systèmes de verger-maraîcher. Thèse de Doctorat à l'Université Toulouse 3 Paul Sabatier (UT3), page 24. 14

Marx, D. (2004). Graph colouring problems and their applications in scheduling. Periodica Polytechnica Electrical Engineering (Archives), 48 :11–16. 47

Méndez-Díaz, I. and Zabala, P. (2010). Solving a multicoloring problem with overlaps using integer programming. Discrete Applied Mathematics, 158(4) :349–354. 47

Metropolis, N., Rosenbluth, M., and Teller, A., T. E. (1953). Equation of state calculation by fast computing machines. J. of Chemical Physics, 21 :1087-1092. 21, 50

Meuwly, F., Ries, B., and Zufferey, N. (2010). Solution methods for a scheduling problem with incompatibility and precedence constraints. Algorithmic Operations Research., 5(2) :75–85. 84

Mosa, M., Hamouda, A., and Marei, M. (2017). Graph coloring and aco based summarization for social networks. Expert Systems with Applications, 74 :115–126. 44

Orden, D., Marsa-Maestre, I., Gimenez-Guzman, J., de la Hoz, E., and Álvarez-Suárez, A. (2019). Spectrum graph coloring to improve wi-fi channel assignment in a real-world scenario via edge contraction. Discrete Applied Mathematics, 263 :243–234. 44

Pareto, V. (1896). Multiobjective optimization tool for engineering design. Cours d'économie politique, 1 et 2. 72

Pinedo, M. (2008). Scheduling : Theory, algorithms, and systems. Prentice Hall Third, 15,16. 23, 30

Ramesh, C., Kamalakannan, R., Karthik, R. Pavin, C., and Dhivaharan, S. (2021). A lot streaming based flow shop scheduling problem using simulated annealing algorithm. Materials Today : Proceedings, 37 :241–244. 52

Sakarovitch, M. (1984). Optimisation combinatoire : méthodes mathématiques et algorithmiques, graphes et programmation linéaire. Paris : Hermann, 1. 15

Schaffer, J. and Grefenstette, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. ICGA International Conference on Genetic Algorithms. Lecture Notes in Computer Science, pages 93–100. 78

Serafini, P. (1992). Simulated annealing for multiple objective optimization problems. Tenth Int. Conf. on Multiple Criteria Decision Making, pages 87–96. 80

Shukri, S., Al-Sayyed, R., and Hudaib, A.and Mirjalili, S. (2021). Enhanced multi-verse optimizer for task scheduling in a cloud computing environments. Expert Systems with Applications, 168 :14–230. 45

Sotskov, Y., Dolgui, A., and Werner, F. (2001). Mixed graph coloring for unit-time job-shop scheduling. International Journal of Mathematical Algorithms, 2(4) :289–323. 45

Sotskov, Y. and Tanaev, V. (1976). Chromatic polynomial of a mixed graph. Vestsi Akademii Navuk BSSR, Ser. Fiz. Mat. Navuk, 6 :20–23. 45

Suppapitnarm, A. and Parks, T. (2001). Simulated annealing : an alternative approach to true multiobjective optimization. Genetic and Evolutionary Computation Conférence. 81

Thevenin, S., Zufferey, N., and Potvin, J. (2013). Tabu search for a preemptive scheduling problem with job incompatibilities. IFAC Proceedings Volumes, 46. 55, 59

Thevenin, S., Zufferey, N., and Potvin, J. (2017). Makespan minimisation for a parallel machine scheduling problem with preemption and job incompatibility. International Journal of Production Research, 55(6) :1588–1606. 4, 6, 47

Thevenin, S., Zufferey, N., and Potvin, J. (2018a). Graph multi-coloring for a job scheduling application. Discrete Applied Mathematics, 234 :218–235. 41

Thevenin, S., Zufferey, N., and Potvin, J. (2018b). Reducibility among combinatorial problems. Complexity of computer computations, 234 :218–235. 44, 47, 55, 56, 59, 60, 61, 93

T'Kinkt, V., N.Monmarché, Tecinet, F., and Laugt., D. (2002). An ant colony optimisation algorithm to solve a 2-machine bicriteria flowshop scheduling problem. European Journal of operational Research., 142 :250–257. 84

Ulungu, E., J., T., Ph., F., and Tuyttens, D. (1999). a tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis, 8 :21–236. 81

Vaidya, J., Bashar, M. K., and Shukla, N. (2021). Using noise to augment synchronization among oscillators. Scientific Reports, 11(1) :1–8. 44

Zheng, Z., Shi, X., He, L. J., H. Wei, S. Dai, H., and Peng, X. (2020). Algorithm with color-centric paradigm on gpu. IEEE Transactions on Parallel and Distributed Systems, 32(1) :160–173. 44

Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization : methods and applications. PhD thesis, Swiss Federal Institute of Technology. 94