Planification et ordonnancement de la maintenance conditionnée par la santé de l'équipement

Ernest Foussard

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : I-MEP2 - Ingénierie - Matériaux, Mécanique, Environnement, Energétique, Procédés, Production
Spécialité : GI - Génie Industriel : conception et production
Unités de recherche : Laboratoire des Sciences pour la Conception, l'Optimisation et la Production de Grenoble et Laboratoire d'Informatique de Grenoble

Planification et ordonnancement de la maintenance conditionnée par la santé de l'équipement

Maintenance planning and scheduling based on equipement health

Présentée par :

Ernest FOUSSARD

Direction de thèse :

Marie-Laure ESPINOUSE PROFESSEURE DES UNIVERSITES, Université Grenoble Alpes	Directrice de thèse
Margaux NATTAF MAITRESSE DE CONFERENCES, Université Grenoble Alpes	Co-encadrante de thèse
Grégory MOUNIÉ MAITRE DE CONFERENCES, Université Grenoble-Alpes	Co-encadrant de thèse

Rapporteurs :

Ayse AKBALIK
MAITRESSE DE CONFERENCES HDR, Université de Lorraine
Stéphane DAUZERE-PERES
PROFESSEUR DES UNIVERSITES, Ecole des Mines de Saint-Etienne

Thèse soutenue publiquement le **21 décembre 2023**, devant le jury composé de :

Marie-Laure ESPINOUSE PROFESSEURE DES UNIVERSITES, Université Grenoble Alpes	Directrice de thèse
Ayse AKBALIK MAITRESSE DE CONFERENCES HDR, Université de Lorraine	Rapporteure
Stéphane DAUZERE-PERES PROFESSEUR DES UNIVERSITES, Ecole des Mines de Saint-Etienne	Rapporteur
Bernard PENZ PROFESSEUR DES UNIVERSITES, Grenoble INP	Président
Safia KEDAD-SIDHOUM PROFESSEURE DES UNIVERSITES, CNAM Paris	Examinatrice

Invités :

Margaux Nattaf
MAITRE DE CONFERENCES, Université Grenoble-Alpes
Grégory Mounié
MAITRE DE CONFERENCES, Université Grenoble-Alpes

Remerciements

Ce travail a bénéficié d'aides de L'État gérées par l'Agence Nationale de la Recherche au titre du programme «France 2030» (ANR-15-IDEX-0002) et du LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

⋆ ⋆ ⋆

Ces trois années de thèse ont été très riches en rencontres et en moments de partage, aussi bien sur le plan professionnel que sur le plan personnel. Cette aventure était loin d'être solitaire, vous êtes nombreuses et nombreux à avoir contribué, chacun à votre manière, à ce succès. Je souhaiterais donc prendre le temps ici de remercier toutes celles et tous ceux qui m'ont accompagné jusqu'ici.

J'aimerais commencer par remercier mon jury, Bernard Penz, Safia Kedad-Sidhoum et tout particulièrement les rapporteur·e·s Ayse Akbalik et Stéphane Dauzère-Pérès. J'ai grandement apprécié vos retours, la qualité de nos échanges et vos propositions d'améliorations, qui ont, je crois, permis de grandement améliorer la qualité de mon manuscrit.

L'aventure a commencé en février 2020, lorsque j'ai commencé mon stage de master à Marie-Laure et Margaux, puis Grégory nous a rejoints six mois plus tard lorsque j'ai commencé ma thèse. Je souhaiterais vous remercier tous les trois pour votre confiance renouvelée pendant ces quatre ans. Marie-Laure, merci d'avoir accepté de diriger ma thèse, et de m'avoir conduit vers ce succès. Je te suis très reconnaissant pour ta confiance, l'autonomie que tu m'as laissé et pour ton professionnalisme. Merci de m'avoir accompagné pour mes premiers pas dans le monde de la recherche. Margaux, j'ai beaucoup apprécié travaillé avec toi. Merci pour ta disponibilité, de m'avoir accompagné quand il fallait mettre les mains dans le cambouis. Je garderai un très bon souvenir des moments de galère devant mes programmes dynamiques et mes preuves, mais aussi des super moments une fois le travail terminé : au sun, en conf... je te souhaite le meilleur pour la suite. Grégory, sans toi les réunions auraient sans doute été (beaucoup) plus courtes, mais aussi beaucoup plus ennuyeuses. J'ai beaucoup apprécié ton regard différent sur les choses, ton humanité et ta bienveillance.

J'ai également une pensée pour Nabil et Christophe, qui ont suivi la progression de ma thèse dans le cadre du CSI.

J'aimerais aussi remercier ceux avec qui j'ai collaboré pour obtenir certains des résultats que je présente ici : Gérémi et Vincent. Merci aussi à tous ceux qui au détour d'une discussion dans un couloir en conférence m'ont permis de découvrir de nouvelles pistes et de nouvelles idées.

Je remercie également mes collègues au G-SCOP et au LIG pour leur accueil, et tout spécialement les collègues des équipes GROG (ex-ROSP) et Datamove pour m'avoir accompa-

gné et formé pendant ces trois ans. Merci pour les discussions et les échanges passionnants non-seulement sur la RO, mais aussi sur tout le reste. Merci aussi aux collègues qui m'ont fait confiance et ont accompagné mes premiers pas en tant qu'enseignant, celles et ceux avec qui j'ai passé de nombreuses heures en pause café et au sun, celles et ceux avec qui j'ai partagé un bureau Akash, Adrien, Astor, Camille, Félix, Marco, Sofi et Victor. Et une pensée particulière pour celles et ceux qui sont devenus des ami·e·s précieux·ses et avec qui les chouettes moments se prolongent au-delà des portes et des horaires du labo Hugo, Maëva, Sami, Suzanne, Thomas et Ugo.

J'ai la chance dans ma vie quotidienne d'être merveilleusement bien entouré par mes ami·e·s, celles et ceux avec qui j'ai fait mes études, celles et ceux avec qui j'ai dansé, celles et ceux avec qui vécu... et celles et ceux qui sont toujours là après tant de temps : merci Abdel, Andréa, Ben, Benjamin, Brian, Camille, Charlie, Corentin, Dylan, Edouard, Enguerran, Floriane, Guilhaume, Julien, Lucie, Lucille, Manon, Pavan, Raoul, Romain, Théo, Titus, Vivien et aussi tous les autres que j'oublie. Je voulais tout particulièrement remercier trois ami·e·s particulier·e·s qui ont marqué ma vie ces trois dernières années. Évane, merci pour la richesse de nos moments partagés, les danses et les discussions passionnantes jusqu'à des heures beaucoup trop tardives. Merci de m'avoir écouté, soutenu, aidé à découvrir et explorer de nouveaux chemins. Thed, merci pour cette amitié sincère et si unique, pour ta gentillesse et pour les fous rires. Merci aussi d'avoir toujours répondu présent pour me soutenir dans les moments difficiles. Et puis Manon, pendant ces trois ans, tu auras été à la fois ma coloc, ma collègue (même si c'était pas prévu), ma confidente et surtout une vraie amie. Imaginait-on seulement dans quoi on s'embarquait au moment de commencer nos thèses et la coloc ? Merci d'avoir toujours été là, de m'avoir aidé à me relever lorsque j'étais au plus bas. Tu as été la petite étincelle qui a déclenché tant de choses, qui m'a permis de franchir tant d'obstacles qui me paraissaient insurmontables. Merci infiniment.

Merci enfin à ma famille, mes grands-parents, oncles et tantes, cousins, cousines, ma sœur et mes parents, venus en nombre à ma soutenance pour me soutenir. Je sais que je ne serais pas là où j'en suis aujourd'hui sans vous, que malgré la distance je peux toujours compter sur votre soutien et je vous en suis très reconnaissant.

Résumé

La problématique de la maintenance est l'un des enjeux majeurs de la gestion des systèmes de production industriels. Une politique de maintenance efficace permet de limiter les pannes et les dysfonctionnements de l'équipement, et ainsi d'en améliorer la disponibilité et le niveau de service. Elle joue également un rôle central dans la durée de vie des équipements et se retrouve ainsi au cœur des schémas économiques visant à allonger la durée de vie et la réemployabilité tels que l'économie circulaire. L'avènement de l'industrie 4.0 et les progrès technologiques récents sur les capteurs ont permis d'établir des modèles de dégradations précis pour l'équipement industriel, ce qui a permis l'émergence de nouvelles stratégies de maintenances, dites prédictives, s'appuyant sur l'évolution de l'état des composants de l'équipement au cours du temps. Dans le cadre de cette thèse, nous nous intéressons plus spécifiquement à la problématique de la planification et de l'ordonnancement de la maintenance en présence d'un indicateur de santé sur les composants de l'équipement.

Dans une première partie, nous proposons un cadre général pour la modélisation et la résolution de problèmes de planification de la maintenance au niveau tactique en présence d'un indicateur de santé des composants de l'équipement. En établissant que la santé des composants de l'équipement peut être traitée comme un potentiel de production pour l'équipement, nous montrons que les problèmes de planification de la maintenance au niveau tactique se ramènent à des variantes des problèmes de dimensionnement de lot, et en déduisons divers résultats de complexité. Nous nous intéressons plus spécifiquement à un problème bi-critère faisant intervenir une contrainte de ressources dépendante de la santé des composants de l'équipement au cours du temps et proposons une modélisation en programmation linéaire en nombres entiers multi-objectif. Une méthodologie d'aide à la décision, permettant de fournir un ensemble restreint de solutions diversifiées à un preneur de décision est ensuite proposée.

Dans la seconde partie de ce manuscrit, nous étudions l'ordonnancement conjoint de la production et de la maintenance basé sur la santé des composants de l'équipement. Une nouvelle extension du système de notation et de classification des problèmes d'ordonnancement est proposée afin d'y intégrer les problèmes d'ordonnancement conjoint de la production et de la maintenance. Nous proposons une cartographie de la complexité pour le problème étudié et ses variantes, une modélisation en programmation linéaire en nombres entiers, et nous étudions les propriétés des problèmes de minimisation du délai maximum et du délai moyen. Dans un second temps, nous introduisons une contrainte de seuils de santé acceptables pour l'exécution des tâches et étudions un cas particulier du problème se ramenant à une variante du problème de bin-packing. Plusieurs approches de résolution basées sur la programmation linéaire en nombres entiers sont alors présentées et comparées.

Abstract

Maintenance is one of the crucial issues in the management of industrial production systems. Effective maintenance policies limit equipment breakdowns and malfunctions, and thus improve availability and level of service. It also plays a central role in the lifespan of equipment, and is thus at the heart of economic schemes aiming at extending lifespan and re-usability, such as circular economy. The rise of 4.0 Industry and recent technological advances in sensors have made it possible to establish precise degradation models for industrial equipment, enabling the emergence of new maintenance strategies, known as predictive maintenance, based on the evolution of the condition of its components over time. In this thesis, we focus more specifically on maintenance planning and scheduling problems in the presence of a health indicator on the components of the equipment.

In the first part, we propose a broad framework for modeling and solving maintenance planning problems at the tactical decision level in the presence of a health indicator on the components of the equipment. Observing that health could be treated as a production potential for the equipment, we show that these problems can be reduced to variants of lot-sizing problems and present various theoretical results. More specifically, we focus on a bi-criteria problem involving a resource constraint dependent on health of the components of the equipment over time, and propose a multi-objective integer linear programming model. Then, a decision-support methodology is proposed, providing a decision-maker with a restricted set of diversified solutions.

In the second part of this manuscript, we study joint production and maintenance scheduling based on the health of the components of the equipment. A new extension of the classical scheduling notation and classification system is proposed to incorporate joint production and maintenance scheduling problems. We propose a complexity mapping for the studied problem and its variants, an integer linear programming model, and study the properties of maximum and average completion time minimization problems. Secondly, we introduce a constraint of acceptable health thresholds for task processing, and study a special case of the problem as a variant of the classical bin-packing problem. Several solution approaches based on integer linear programming are then presented and compared.

Table des matières

Introduction

La problématique de la maintenance est l'un des enjeux majeurs de la gestion des systèmes de production industriels. Une politique de maintenance efficace permet de limiter les pannes et les dysfonctionnements de l'équipement, et ainsi d'en améliorer la disponibilité et le niveau de service. Elle joue également un rôle central dans la durée de vie des équipements et se retrouve ainsi au cœur des schémas économiques visant à allonger la durée de vie et la réemployabilité tels que l'économie circulaire.

Selon les associations européennes de normalisation, la maintenance désigne *"l'ensemble de toutes les actions techniques, administratives et de management durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise."* (NF EN 13306 X 60-319). Cette définition est très large et recouvre des réalités très diverses. La maintenance désigne aussi bien la simple opération de nettoyage ou d'inspection d'un équipement que le remplacement d'une partie de celui-ci.

Pendant longtemps dans le milieu industriel, la maintenance consistait essentiellement à réparer l'équipement lorsqu'il tombait en panne. Toutefois, les évolutions technologiques au cours des dernières décennies et les transformations de l'appareil de production qui en ont découlé ont rendu à la fois possible et indispensable de repenser ce paradigme au profit d'approches plus sophistiquées visant à anticiper la panne et à agir en amont. Cette évolution s'est accélérée récemment avec l'avènement de l'industrie 4.0 et les dernières innovations sur les capteurs, qui ont permis l'émergence de stratégies de maintenance prédictive sophistiquées basées sur un suivi fin de l'évolution de l'état de l'équipement au cours du temps.

Dans cette thèse, nous nous intéressons à la problématique de l'optimisation des plannings de maintenance dans ce nouveau contexte, où l'état de l'équipement peut être décrit par un indicateur de santé. Nous abordons de nouveaux problèmes à des niveaux de décision divers liés à cette thématique et proposons des méthodes de modélisation et de résolution issues du domaine de l'optimisation mathématique et de l'algorithmique.

Organisation du manuscrit

Cette thèse s'articule autour de six chapitres.

Dans le chapitre 1, nous introduisons le contexte global dans lequel se situent les problèmes étudiés dans le cadre de cette thèse.

Les quatre chapitres suivants se répartissent dans deux parties correspondant aux deux problèmes que nous traitons.

Dans la partie I, nous nous intéressons à un problème de planification de la maintenance bi-critère avec prise en compte d'un indicateur de santé de l'équipement au niveau tactique

de décision.

Le chapitre 2 introduit le problème et le positionne par rapport à l'état de l'art de la planification tactique de la production et de la maintenance. Nous proposons une approche originale, établissant des liens avec les problèmes de lot-sizing classiquement utilisés pour la planification de la production. Les principaux résultats théoriques de modélisation et de complexité y sont présentés.

Le chapitre 3 se concentre plus spécifiquement sur la dimension bi-critère et la mise en place d'une méthodologie d'aide à la décision pour la résolution pratique du problème. Nous expérimentons dans un premier temps une approche consistant à imposer une contrainte de budget sur un des deux objectifs afin de se ramener à un problème mono-objectif. Dans un second temps, nous nous intéressons au calcul de fronts de Pareto et en étudions les propriétés. Une procédure d'aide à la décision ad hoc est ensuite dérivée permettant de fournir efficacement un ensemble de solutions satisfaisantes au preneur de décision. Les différentes méthodes sont implémentées et testées sur des instances de taille réaliste générées aléatoirement.

La partie II concerne la planification conjointe de la production et de la maintenance au niveau opérationnel de décision basé sur un indicateur de santé de l'équipement.

Dans le chapitre 4, nous définissons le problème et étudions son positionnement dans la littérature de l'ordonnancement conjoint de la production et de la maintenance. Nous étudions les différentes propriétés et caractéristiques du problème et nous proposons un panorama de la complexité et de l'approximabilité des différentes variantes du problème, ainsi qu'une modélisation mathématique.

Dans le chapitre 5, nous explorons plus en détail une variante spécifique du problème se rapprochant des problèmes de bin-packing. Différentes méthodes de la littérature basées sur la programmation linéaire en nombres entiers sont adaptées afin de prendre en compte les spécificités du problème, puis comparées expérimentalement sur un jeu d'instance généré aléatoirement.

Un résumé des contributions et les principales perspectives de recherche sont présentées en conclusion de chacune des deux parties.

Enfin, le chapitre 6 conclut cette thèse avec un résumé plus global des travaux présentés et une présentation des perspectives plus générales concernant l'ensemble des travaux.

Chapitre 1

Contexte des travaux

Sommaire

Ce premier chapitre introductif vise à introduire le contexte général des travaux présentés dans cette thèse. Dans un premier temps, nous nous intéressons au contexte bibliographique dans lequel s'inscrivent ces travaux relativement à la thématique de l'optimisation de la maintenance. Nous introduisons ensuite la problématique générale et séparons le problème en sous-problèmes correspondant aux différents niveaux de décision. Dans un second temps, nous introduisons les outils méthodologiques principaux utilisés tout au long de la thèse.

1.1 Contexte du problème

La part de la maintenance dans les coûts de production n'a cessé d'augmenter au cours des dernières décennies, et peut représenter entre 15% et 70% des coûts selon le type d'industrie (Al-Turki et al., 2014). Zio et Compare (2013) énumèrent les principaux facteurs qui expliquent ce phénomène : mécanisation et sophistication de l'équipement, le développement de l'externalisation de la production, normes de sécurité de plus en plus contraignantes et l'augmentation du besoin de réactivité, liée notamment à l'ouverture du marché de l'électricité et au développement de la production à flux tendu. Cette tendance est amenée à se confirmer dans les années à venir, avec le développement, notamment dans le secteur de l'énergie, d'industries basées sur des technologies ayant des besoins de maintenance importants. Les transformations récentes de l'appareil de production liées à l'Industrie 4.0 ont accéléré l'évolution des stratégies de maintenance, avec l'émergence de nouveaux concepts et terminologies telles que l'e-maintenance ou la maintenance intelligente (Roda & Macchi, 2021).

L'optimisation des politiques de maintenance est ainsi devenue de plus en plus incontournable pour les industriels dans le but d'améliorer la compétitivité de leur industrie (Zio & Compare, 2013).

1.1.1 Optimisation de la planification de la maintenance

Le domaine de l'optimisation de la maintenance a fait l'objet d'une revue de littérature récente par de Jonge et Scarf (2020).

Il n'existe pas de consensus dans la littérature pour la classification des stratégies de maintenance. Certains auteurs en distinguent trois grandes familles : la maintenance corrective, la maintenance préventive et la maintenance prédictive (Zhao et al., 2022). La maintenance corrective ou curative, consiste à effectuer des maintenances lorsque l'équipement subit une panne. On distingue au sein de cette famille la maintenance corrective d'urgence effectuée immédiatement lorsque la panne survient, et la maintenance corrective planifiée, qui implique une période d'inactivité entre le moment où la panne survient et la réparation. Les stratégies de maintenance corrective pures sont coûteuses et impactent significativement l'activité de l'équipement. Pour cette raison, elles ont progressivement été délaissées au profit d'autres stratégies plus sophistiquées.

Les deux autres familles de stratégie de maintenance sont regroupées par certains auteurs sous le terme de maintenance proactive, par opposition à la maintenance corrective qui est effectuée en réaction à une panne. La maintenance préventive consiste à planifier des opérations de maintenance en amont en se basant sur les connaissances que l'on dispose à propos de l'équipement, de la fréquence des pannes au cours du temps... l'objectif de cette

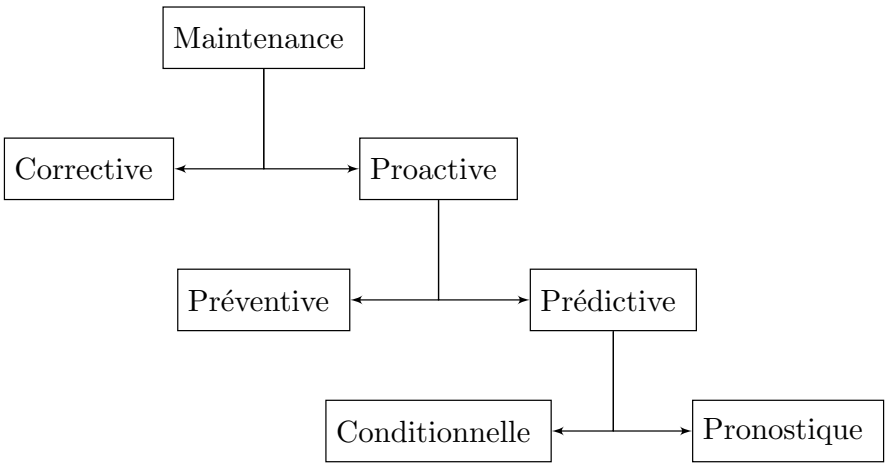


FIGURE 1.1 – Les différentes stratégies de maintenance

stratégie de maintenance est de minimiser le risque de panne afin d'éviter les pertes d'activités ou les surcoûts liés. On retrouve en particulier dans cette famille les politiques de maintenance périodiques, basées sur l'âge, séquentielles... La maintenance prédictive quant à elle consiste à suivre l'évolution de l'état des composants de l'équipement à l'aide de technologies le permettant (notamment les capteurs) dans le but d'adapter la politique de maintenance. On distingue notamment la maintenance conditionnelle, qui consiste à intervenir lorsqu'un ensemble de conditions sur les mesures fournies par les capteurs ne sont plus satisfaites, dans le but de n'effectuer les maintenances qu'uniquement lorsqu'elles sont nécessaires (Y. Li et al., 2020 ; Prajapati et al., 2012 ; Quatrini et al., 2020), et les approches type Prognostics and Health Management (PHM) basées sur les données qui consistent à faire des prédictions sur l'évolution de l'état de l'équipement (Baur et al., 2020). Pour le cas de la maintenance préventive comme celui de la maintenance prédictive, un des principaux défis concerne les données, le coût et la complexité de la mise en place de ces systèmes : la collecte de données utilisables sur les fréquences de panne d'un équipement est un processus long qui peut s'étaler sur des années, tandis que les technologies nécessaires à la mise en place d'une politique de maintenance prédictive sont souvent très onéreuses et la modélisation de la dégradation à partir de ces données est un véritable défi (Ahmad & Kamaruddin, 2012 ; Hassankhani Dolatabadi & Budinska, 2021).

Le diagramme présenté sur la Figure 1.1 résume cette classification.

La littérature de la planification de la maintenance peut également être classée selon la nature du système de production auquel les stratégies de maintenance sont appliquées.

Les systèmes de production subissent en général de la dégradation au cours de leur existence et peuvent être sujets à de la maintenance. En général, les systèmes de production sont complexes et peuvent être décomposés en plusieurs sous-systèmes, pouvant eux même potentiellement être décomposés en d'autres sous-systèmes. En maintenance, on s'intéresse plus spécifiquement aux parties du système qui sont sujettes à la dégradation et à la maintenance.

Ainsi, le terme *composant* désigne un sous-système qui subit de la dégradation, fait l'objet individuellement de maintenance, et qui ne peut pas être divisé en d'autres sous-composants (de Jonge & Scarf, 2020). On distingue alors les systèmes mono-composants des systèmes multi-composants. Il faut noter toutefois que selon le point de vue et la stratégie de maintenance, un même système peut être l'un ou l'autre. Dans un système multi-composant, il peut exister des dépendances plus ou moins marquées entre les différents composants. En particulier, si une panne ou une maintenance sur un seul composant entraîne l'interruption du fonctionnement de l'intégralité du système, on dit que les composants sont connectés *en série*. Réciproquement, si les autres composants ne sont pas impactés, on parle de composants connectés *en parallèle*. Par exemple, en théorie de l'ordonnancement, un événement se produisant sur une seule des unités d'un système de type flow-shop impacte l'ensemble du système, ce qui correspond à la définition de système en série. Au contraire, dans le cas de l'ordonnancement à machines parallèles, chacune des machines peut être considérée comme un des composants du système, tous connectés en parallèle. Les systèmes en série et en parallèle sont les systèmes multi-composants les plus simples, mais il existe dans la littérature une multitude de travaux sur des systèmes beaucoup plus complexes. La revue de Olde Keizer et al. (2017) sur les politiques de maintenances conditionnelles s'intéresse spécifiquement aux systèmes multi-composants et propose une classification des travaux selon la nature des dépendances entre les composants.

On distingue également différentes manières de décrire la dégradation des composants. Dans un *espace de détérioration discret*, il existe un nombre fini d'états dans lequel le composant peut se situer. La modélisation par un espace à deux états, respectivement *fonctionnel* et *en panne*, est la plus basique et correspond généralement aux stratégies de maintenance corrective pures ou à certaines stratégies de maintenance préventive. Les espaces à trois états introduisent un état intermédiaire correspondant à un composant défectueux. Les stratégies de maintenance préventive *delay-time models* (W. Wang, 2012) sont très courantes dans la littérature et s'appuient sur ce type de modélisation : les composants défectueux peuvent être détectés en effectuant des inspections et réparés avant que la panne ne survienne.

Au contraire, dans un *espace de détérioration continu*, la dégradation peut prendre n'importe quelle valeur dans un intervalle. Les approches impliquant un grand nombre d'états sont rendues possibles par les innovations technologiques qui permettent un suivi plus précis de l'état de l'équipement.

Les opérations de maintenance peuvent être également de plusieurs types. Une *maintenance parfaite* permet de remettre un composant dans un état équivalent à son état d'origine (*as-good-as-new*). C'est par exemple le cas lorsque le composant concerné est remplacé par un composant neuf. Par opposition, la *maintenance imparfaite* ne remet pas dans l'état d'origine le composant et correspond par exemple à une réparation. Un cas particulier de maintenance imparfaite est la *réparation minimale*, qui remet en marche un composant en panne sans toutefois améliorer son état (*as-bad-as-old*).

Dans le cadre de cette thèse, nous nous intéressons au cas d'un système multi-composants en série, avec un espace de dégradation continu, sujet à des maintenances parfaites et imparfaites. Nous nous intéressons à des stratégies de maintenance prédictive, basée sur un indicateur prognostique particulier appelé *santé de l'équipement*, principalement utilisé dans le domaine de la production de semi-conducteurs. A. Chen et Wu (2007) présentent une méthodologie de calcul pour cet indicateur dans un contexte de dégradation markovienne et dérivent une

stratégie de maintenance prédictive. Cet indicateur a été intégré dans plusieurs travaux en planification de la production. Kao et al. (2018) étudient un problème d'ordonnancement de la production où la santé de l'équipement permet de déterminer si une tâche peut-être exécutée sans risque. Plus récemment, Penz et al. (2023) se sont intéressés à un problème d'ordonnancement conjoint de la production et de la maintenance sur un système multi-composant en parallèle où des seuils de santé minimum doivent être respectés pour permettre l'exécution d'une tâche.

1.1.2 Définition du problème global

La problématique générale de cette thèse peut être résumée par la question suivante : " Comment optimiser une politique de maintenance prédictive basée sur un indicateur de santé à l'échelle d'un système de production ? " Cette thématique est très vaste et peut être abordée avec des outils provenant de champs disciplinaires variés. Nous nous intéressons plus spécifiquement à l'application, quand elle est possible, des techniques issues de l'optimisation mathématique et de l'algorithmique pour l'étude de ce problème.

Les travaux préliminaires présentés dans Foussard (2021) ont permis de mettre en avant la nécessité de traiter de manière séparée différents sous-problèmes correspondant à des temporalités différentes. Cette pratique est courante dans la gestion des systèmes de production, et se traduit le plus généralement par la division en trois niveaux de décision : stratégique, tactique et opérationnel. Nous présentons leur définition ci-après et indiquons à quoi correspondent ces niveaux de décision pour notre problème.

Niveau stratégique

Mintzberg et al. (1976) définissent une décision stratégique comme une décision importante en termes d'actions effectuées, de ressources engagées et de précédents créés. Il s'agit donc des décisions prises au plus haut niveau d'une organisation, et qui jouent un rôle critique dans sa pérennité et sa survie.

Concernant la planification de la production, directement liée à la question de la maintenance, la décision stratégique concerne le design et la configuration du système sur un horizon de temps de l'ordre de 5 à 10 ans (Díaz-Madroñero et al., 2014).

Selon Tsang (2002), il est possible d'identifier quatre volets concernant la prise de décision au niveau stratégique pour la gestion de la maintenance :

- Question de l'externalisation de la maintenance : définition de ce qui peut être externalisé, définition du contrat de maintenance avec le sous-traitant, gestion des risques liés à l'externalisation ;
- Organisation des infrastructures de maintenance, et en particulier de la main d'œuvre ;
- Choix de la stratégie de maintenance ;
- Mise en place de systèmes de soutien, dont les systèmes permettant la collecte de données relatives à la maintenance.

Pour le cas de notre problème, le niveau stratégique de décision englobe les décisions de long terme sur la gestion de l'équipement (cycle de vie des composants, acquisitions...), la définition d'une stratégie sur la gestion de ressources et des déchets, la gestion du risque stratégique (par exemple : changement de législation, épidémie, guerre...) et les aspects stratégiques concernant

la maintenance, en particulier la mise en place des systèmes de collectes de données et la méthodologie de calcul pour l'indicateur de santé.

Si la question stratégique est essentielle dans la gestion d'un système de production, la plupart des problématiques liées à ce niveau font appel à des champs disciplinaires très éloignés de la recherche opérationnelle et de l'optimisation mathématique.

Pour cette raison, le niveau stratégique de décision ne sera pas traité dans le cadre de cette thèse.

Niveau tactique

Selon Pintelon et Gelders (1992), la planification de la maintenance au niveau tactique consiste à trouver une politique de maintenance optimale, c'est-à-dire trouver la meilleure quantité et la meilleure combinaison d'activités de maintenance permettant de garantir que chaque partie de l'équipement ait le niveau de service et le niveau de disponibilité désiré.

Pour notre problème, l'optimisation du planning de maintenance au niveau tactique consiste d'une part à définir les volumes de maintenance nécessaires pour garantir le niveau de service voulu, et éventuellement satisfaire des contraintes liées à des questions de fiabilité, en s'appuyant sur l'indicateur de santé de l'équipement. De plus, il s'agit aussi de placer ces volumes de maintenances dans le temps de sorte à optimiser la disponibilité de la machine vis-à-vis des besoins.

Dans la partie I de ce manuscrit, nous étudions la planification tactique de la maintenance dans la configuration suivante. On considère un équipement possédant plusieurs composants plusieurs composants, chacun ayant un niveau de santé se dégradant de manière proportionnelle au niveau de production. L'horizon de temps est échantillonné en périodes, et à chaque période l'équipement possède une capacité de production théorique pleinement utilisée par défaut. Il est possible également à chaque période de décider d'effectuer une ou plusieurs maintenances, ce qui permet de régénérer la santé d'un ou plusieurs composants, au détriment de la perte d'une fraction de la capacité de production théorique. De plus, si la santé de l'équipement atteint zéro, l'équipement devient inopérable. On fait l'hypothèse que les maintenances sont mutuellement exclusives, et que par conséquent les pertes de capacité s'additionnent. On cherche alors à minimiser deux objectifs : un objectif de coût composé des coûts directs de maintenance et des coûts d'opportunité liés à la perte de capacité de production, et un objectif générique de consommation de ressources dépendant du niveau de santé de composants lors de la production.

Niveau opérationnel

Le niveau opérationnel concerne des échelles de temps beaucoup plus courtes, de l'ordre du jour et de la semaine, et consiste pour un système de production, à la mise en place en pratique du planning décidé au niveau tactique, en définissant, en assignant et ordonnançant les tâches à effectuer.

La planification de la maintenance au niveau opérationnel consiste à placer les tâches de maintenance dans la séquence des tâches à effectuer et leur affectation à la main d'œuvre. À ce niveau, le planning doit prendre en compte différentes contraintes comme les priorités sur les tâches à effectuer, la disponibilité des équipes de maintenance, des pièces de rechange et

de l'équipement (Pintelon & Gelders, 1992).

Dans la partie II de ce manuscrit, nous étudions un problème d'ordonnancement au niveau opérationnel de la production et de la maintenance. Similairement au problème étudié au niveau tactique, on considère un équipement multi-composant, chaque composant ayant son propre niveau de santé, sur un horizon de temps discrétisé, et la santé des composants ne doit pas descendre en dessous de zéro. Les activités de production sont séparées en tâches, de durées connues et causant une usure fixe sur l'équipement. La maintenance se présente également sous forme de tâches de durées connues, qui permettent de régénérer les composants. Certaines tâches requièrent un niveau de santé minimum au moment de leur traitement. On cherche à trouver des ordonnancements réalisables minimisant des objectifs classiques tels que le délai maximum C_{max} ou le délai moyen $\sum C_i$.

Les deux sous-problèmes que nous traitons dans cette thèse correspondent à deux systèmes de production différents ayant des horizons de dégradation différents : pour le cas tactique, nous nous intéressons à un système pour lequel la dégradation de l'équipement se passe typiquement à l'échelle du mois ou de l'année (moyen-terme) tandis qu'au niveau opérationnel, nous traitons de dégradation à l'échelle du jour ou de la semaine (court-terme).

1.2 Contexte méthodologique

Cette thèse se situe dans le cadre classique de la recherche opérationnelle, et fait intervenir des méthodes notamment issues du champ de l'optimisation mathématique et de l'algorithmique. En particulier, les méthodes et techniques issues de la théorie de la complexité et de la programmation linéaire en nombres entiers mixte sont utilisées tout au long de la thèse. Dans cette section, nous rappelons les principaux concepts, définitions, notations et résultats de ces deux domaines disciplinaires utiles à la compréhension de ce manuscrit. Les méthodes issues d'autres domaines utilisées plus ponctuellement sont présentées directement dans les chapitres concernés.

1.2.1 Théorie de la complexité

La notion de complexité algorithmique est centrale en recherche opérationnelle. La complexité d'un algorithme peut être définie comme l'ordre de grandeur du nombre d'opérations élémentaires nécessaires pour exécuter l'algorithme étant donné la taille de l'entrée. Par exemple, additionner deux entiers demande environ autant d'opérations qu'il y a de décimales dans le plus grand des deux entiers, on parle alors de complexité linéaire, noté $O(n)$, où n est la taille de l'entrée (ici le nombre de décimales). On s'intéresse plus précisément à la complexité dans le pire des cas.

Il apparaît que certains problèmes sont en pratique très faciles à résoudre par ordinateur, y compris si le problème est de grande taille : faire des opérations arithmétiques élémentaires sur deux nombres, même s'ils ont des millions de décimales, est toujours rapide. Trier un jeu de cartes, même s'il en contient des milliers, est un jeu d'enfant pour un ordinateur avec le tri rapide ou le tri fusion. En revanche, pour certains autres problèmes, il semble impossible de trouver des algorithmes efficaces : trouver le meilleur coup à jouer aux échecs, trouver la tournée la plus courte qui passe par un ensemble de villes définies ou encore

répartir équitablement des tâches entre plusieurs personnes. Pour ces problèmes-là, le meilleur algorithme connu a une complexité exponentielle, et il est vraisemblable qu'on ne trouvera jamais d'algorithmes efficaces permettant de les résoudre. La complexité d'un problème est justement définie comme la complexité du meilleur algorithme permettant de le résoudre. La théorie de la complexité est l'étude de la complexité des problèmes et consiste plus précisément à établir une hiérarchie de difficulté entre les problèmes. En recherche opérationnelle, l'étude de la complexité des problèmes est utile puisqu'elle permet de définir quel type de méthode est plus adaptée pour la résolution d'un problème.

On se limitera ici aux deux classes de complexité P et NP. Les définitions, notions et résultats présentés dans cette section peuvent être retrouvés de manière beaucoup plus détaillée dans l'ouvrage de référence de la théorie de la complexité : *Computers and Intractability ; A Guide to the Theory of NP-Completeness* de Garey et Johnson (1990).

Problèmes de décision, problèmes d'optimisation

Afin de définir ce qu'est la complexité d'un problème, il faut déjà commencer par définir ce qu'est un problème. Un *problème* peut être défini comme une question générale qui peut être accompagnés de paramètres. Une *instance* d'un problème est un sous-problème où les paramètres ont été fixés. Par exemple, on peut définir le problème de voyageur de commerce comme suit : " Étant donné un graphe complet à n sommets muni de poids sur les arêtes, trouver un cycle de poids minimum passant par tous les sommets." Une instance de ce problème peut être par exemple, de trouver un tel cycle pour le graphe dont les sommets sont Paris, Lyon, Marseille, Grenoble et Nantes, et où les poids sont les distances à vol d'oiseau entre les villes. On peut utiliser le nombre de sommets pour définir la taille de l'instance, en l'occurrence 5 pour notre instance.

On distingue en particulier les *problèmes de décision*, où seules deux réponses sont possibles : oui ou non, et les *problèmes d'optimisation*, où la question est de trouver une solution qui minimise un objectif. Pour un problème de décision $\mathcal{D}$, on utilise habituellement la notation $Y_{\mathcal{D}}$ pour désigner l'ensemble des instances de $\mathcal{D}$ qui admettent pour réponse "oui". Le problème du voyageur de commerce tel que défini ci-dessus est un problème d'optimisation, puisque la question est de trouver un cycle qui minimise le poids total.

Les problèmes d'optimisation sont très liés aux problèmes de décision : tout problème d'optimisation peut-être résolu en combinant une dichotomie avec la résolution d'un problème de décision spécifique, appelé problème de décision associé. Pour le problème de voyageur de commerce, le problème de décision associé peut être défini de la manière suivante : " Étant donné un graphe complet à n sommets muni de poids sur les arêtes, existe-t-il un cycle passant par tous les sommets de poids inférieur à un entier K". Ici, K est une donnée du problème. Le problème d'optimisation peut être résolu alors en résolvant de manière successive le problème de décision associé pour différentes valeurs de K, déterminées par dichotomie.

Les classes de problèmes P et NP

Il existe une très large variété de classes de complexité permettant de trier les problèmes de décision. En recherche opérationnelle de manière générale, on s'intéresse habituellement à deux classes de complexité particulières : P pour "Polynomial", et NP pour "Non-deterministic

Polynomial". Par souci de simplicité, nous n'utiliserons pas ici les définitions très formelles utilisant le formalisme des machines de Turing, mais plutôt les définitions un peu moins précises habituellement utilisées par les praticiens de la théorie de la complexité. Nous laissons également de côté les questions d'encodage. Pour plus détails, nous renvoyons le lecteur vers le livre de Garey et Johnson (1990).

Définition 1.2.1. *P (Garey & Johnson, 1990)*
Un problème de décision $\mathcal{D}$ appartient à la classe P s'il existe un algorithme polynomial résolvant $\mathcal{D}$.

La classe P est donc la classe des problèmes de décision pouvant être résolus par des algorithmes polynomiaux.

Définition 1.2.2. *NP (Garey & Johnson, 1990)*
Un problème de décision $\mathcal{D}$ appartient à la classe NP s'il existe un algorithme non-déterministe polynomial résolvant $\mathcal{D}$. De manière moins formelle, $\mathcal{D} \in NP$ si les solutions de $\mathcal{D}$ sont vérifiables en temps polynomial.

La définition de la classe NP est moins intuitive que pour P. Informellement, la classe NP est la classe de l'ensemble des problèmes pour lesquels si on nous donne une preuve (généralement appelée *certificat*) qu'une instance a pour réponse "oui", alors il est possible de vérifier cette preuve en temps polynomial.

Si on reprend notre instance exemple pour le problème de voyageur de commerce, et que l'on fixe la longueur maximale du cycle à 2000 km. Si la solution Paris-Lyon-Grenoble-Marseille-Nantes nous est fournie comme certificat, il suffit juste d'additionner les distances pour constater qu'un cycle de 1736 km existe, et donc que la réponse à la question de l'instance est bien "oui". Pour un graphe à n sommets, ce procédé de vérification nécessite $n+1$ additions, ce qui est bien polynomial. Le problème de décision associé au problème de voyageur de commerce appartient bien à NP.

Il est souvent facile d'observer qu'un problème appartient à NP, néanmoins les preuves formelles sont lourdes et présentent peu d'intérêt. Pour cette raison, les preuves d'appartenance à NP pour les problèmes de décision considérés dans ce manuscrit ne sont pas présentées et on se contentera de les admettre.

À noter que les classes P et NP concernent les seuls problèmes de décision. Un problème d'optimisation ne peut pas appartenir à P ou à NP, même si son problème de décision associé y appartient.

On peut facilement vérifier que $P \subseteq NP$: en effet, si un problème appartient à P, il suffit de le résoudre en temps polynomial pour vérifier qu'une instance est une instance "oui". L'autre sens de l'inclusion est encore une question ouverte à ce jour. La plupart des chercheurs s'accordent toutefois sur la conjecture $P \neq NP$, tant le contraire semble invraisemblable.

Si P est différent de NP, alors la classe P désigne les problèmes qui peuvent être facilement résolus par un ordinateur, tandis que les problèmes de $NP \setminus P$ désignent les problèmes de la classe NP qui sont difficiles à résoudre. Néanmoins, tant que cette conjecture n'aura pas été prouvée, il sera impossible de prouver l'appartenance d'un problème à $NP \setminus P$. Les résultats de la théorie de la complexité concernant la supposée appartenance d'un problème à $NP \setminus P$ sont ainsi toujours conditionnés par l'hypothèse que $P \neq NP$.

Pour prouver l'appartenance d'un problème à $NP \setminus P$, il suffit de transformer le problème en un autre problème dont on sait qu'il appartient à $NP \setminus P$ en utilisant une transformation spécifique, appelée réduction polynomiale.

Définition 1.2.3. *Réduction polynomiale (Garey & Johnson, 1990)*
Une application $f : \mathcal{D}_1 \to \mathcal{D}_2$ est une réduction polynomiale du problème de décision $\mathcal{D}_1$ vers le problème $\mathcal{D}_2$ est une réduction polynomiale si :
— *f peut être calculée par algorithme polynomial*
— *Pour toute instance I de $\mathcal{D}_1$, $I \in Y_{\mathcal{D}_1}$ si et seulement si $f(I) \in Y_{\mathcal{D}_2}$*

La réduction polynomiale a pour propriétés notables de définir une relation d'ordre partiel sur l'ensemble des problèmes de décision et de conserver les ensembles P et $NP \setminus P$.

Si deux problèmes sont mutuellement réductibles l'un dans l'autre, on parle d'*équivalence polynomiale*. L'équivalence polynomiale est une relation d'équivalence sur l'ensemble des problèmes de décision et permet de définir des classes d'équivalence. En particulier, P est une classe d'équivalence pour la relation d'équivalence polynomiale.

Problèmes NP-difficiles

Les notions d'équivalence et de réduction polynomiales sont au cœur de la théorie de la NP-complétude.

Cook (1971) a prouvé l'existence de problèmes dans la classe NP tels que tous les problèmes de la classe NP admettent une réduction polynomiale vers ces problèmes. Ces problèmes sont appelés problèmes NP-complets et forment une classe d'équivalence au sens de l'équivalence polynomiale.

On utilisera principalement les deux définitions suivantes :

Définition 1.2.4. *Problème NP-difficile (Van Leeuwen, 1991)*
Un problème Π est NP-difficile si et seulement si tout problème de décision $\mathcal{D}$ de la classe NP admet une réduction polynomiale vers Π

Définition 1.2.5. *Problème NP-complet (Van Leeuwen, 1991)*
Un problème de décision $\mathcal{D}$ est NP-complet si et seulement si :
— *$\mathcal{D} \in NP$*
— *$\mathcal{D}$ est NP-difficile*

On notera donc que les problèmes d'optimisation n'étant pas des problèmes de décision, ils ne peuvent pas être NP-complets. Ils peuvent être néanmoins être NP-difficiles, et dans ce cas, leurs problèmes de décision associés sont NP-complets, et réciproquement.

Les problèmes NP-complets peuvent être ainsi considérés comme les problèmes les plus difficiles de la classe NP, et à moins que $P = NP$, ces problèmes et les problèmes d'optimisation qui y sont reliés, ne peuvent pas être résolus par des algorithmes polynomiaux.

Pour prouver qu'un problème est NP-difficile, il suffit d'effectuer une réduction polynomiale depuis un autre problème NP-difficile. Depuis 1971, de très nombreux problèmes ont été prouvés NP-complets ou NP-difficiles. Une liste non-exhaustive peut être trouvée dans le livre de Garey et Johnson (1990).

Les problèmes NP-complets n'admettent pas a priori d'algorithmes polynomiaux permettant de les résoudre de manière exacte. Cependant, certains de ces problèmes admettent des algorithmes dont la complexité peut s'écrire comme un polynôme de la taille du problème et de ses paramètres numériques. Ces algorithmes sont appelés *algorithmes pseudo-polynomiaux*, et les problèmes NP-complets pouvant être résolus par de tels algorithmes sont dits *NP-complets au sens faible*, et sont, sous certaines conditions, relativement faciles à résoudre par ordinateur. Dans le cas contraire, on parle de *problèmes NP-complets au sens fort*. Pour prouver qu'un problème est NP-complet au sens fort, on peut effectuer une réduction polynomiale depuis un autre problème NP-complet au sens fort. La réduction doit toutefois vérifier quelques hypothèses supplémentaires. On parle alors de *réduction pseudo-polynomiale*. Le terme peut porter à confusion, puisqu'il s'agit bien d'une réduction polynomiale avec des restrictions supplémentaires.

Étant donné la complexité du sujet, nous n'entrons pas plus en détail ici et renvoyons le lecteur vers Garey et Johnson (1990) pour plus de détails.

Dans cette thèse, afin de ne pas alourdir plus que nécessaire les preuves de complexité, nous admettrons que les réductions sont pseudo-polynomiales dans les cas triviaux.

Approximabilité et compétitivité

Nous nous intéressons également à deux autres notions de théorie de la complexité liées à la problématique de la NP-complétude, concernant plus spécifiquement les problèmes d'optimisation. Pour une instance d'un problème d'optimisation I, on note $A(I)$ la valeur d'objectif renvoyée par un algorithme I, et $OPT(I)$ la valeur de l'optimum.

Définition 1.2.6. *Un algorithme A est une ρ-approximation pour un problème d'optimisation Π si et seulement si $\forall I \in \Pi,\ A(I) \leq \rho OPT(I)$*

Les algorithmes d'approximation sont particulièrement intéressants lorsqu'ils existent puisqu'ils permettent d'apporter des solutions approchées avec une garantie par rapport à l'optimum en temps polynomial pour certains problèmes NP-complets.

La notion de compétitivité se rapproche de la notion d'approximabilité, mais concerne plus spécifiquement les algorithmes online. Un *algorithme online* est un algorithme qui traite les données d'entrée au fur et à mesure de la résolution, contrairement aux algorithmes offline, pour lesquels les données d'entrées doivent être entièrement disponibles dès le début. Les algorithmes online n'apportent généralement pas la solution optimale au problème, puisque des décisions sont prises avant même que l'intégralité de l'instance soit connue. On définit alors la compétitivité de la manière suivante :

Définition 1.2.7. *Un algorithme online A est ρ-compétitif pour un problème d'optimisation Π si et seulement si $\forall I \in \Pi,\ A(I) \leq \rho OPT(I)$*

Similairement aux algorithmes d'approximation, les algorithmes online sont généralement des algorithmes polynomiaux rapides et fonctionnent même quand les données de l'instance ne sont pas entièrement disponibles dès le début, ce qui correspond à de nombreuses situations réelles. La notion de compétitivité permet de fournir une garantie sur la qualité des solutions obtenues à l'aide de ces algorithmes.

1.2.2 Programmation linéaire en nombres entiers

La programmation linéaire est l'un des outils de modélisation les plus utilisés dans le champ de la recherche opérationnelle. Elle consiste à exprimer les différentes décisions possibles à l'aide de variables réelles (ou entières dans le cas de la programmation linéaire en nombres entiers), les objectifs à optimiser comme des combinaisons linéaires de ces variables et les contraintes comme des inégalités sur des combinaisons linéaires de variables.

Nous rappelons ici quelques bases, les principales définitions et les principaux résultats théoriques utiles à la compréhension de la suite du manuscrit. Pour ce faire, nous nous appuyons sur un des ouvrages de référence du domaine, *Theory of Linear and Integer Programming* par Schrijver (1998).

Programmation linéaire

Définition 1.2.8. *Programme linéaire*

Un programme linéaire est un problème d'optimisation pouvant s'écrire de la manière suivante :

$$\begin{aligned} min.\quad & c^\top x && (1.1)\\ s.\ t.\quad & Ax \geq b && (1.2)\\ & x \in \mathbb{R}^n_+ && (1.3) \end{aligned}$$

où $c \in \mathbb{R}^N$ et $b \in \mathbb{R}^M$ sont des vecteurs réels, et A est une matrice réelle de dimension $N \times M$.

Les éléments du vecteur x sont les variables de décision du problème. L'équation (1.1) est l'objectif et l'équation (1.2) définit les contraintes du problème.

Si un vecteur x_0 vérifie l'inégalité $Ax_0 \geq b$, on dit que x_0 est une solution réalisable du programme linéaire. De plus, si x_0 minimise la valeur de l'objectif, on dit que x_0 est une solution optimale du programme linéaire.

Bien qu'il existe des algorithmes polynomiaux pour la résolution des programmes linéaires, la méthode la plus populaire pour leur résolution est la méthode du simplexe, exponentielle dans le pire des cas, mais de complexité polynomiale en moyenne et très rapide en pratique.

Dual d'un programme linéaire

Définition 1.2.9. *Programme linéaire dual*

Soit P un programme linéaire tel que défini dans la définition 1.2.8. Le dual D de P est le programme linéaire défini comme suit :

$$\begin{aligned} max.\quad & b^\top y && (1.4)\\ s.\ t.\quad & A^\top y \leq c && (1.5)\\ & y \in \mathbb{R}^n_+ && (1.6) \end{aligned}$$

P est appelé programme primal et est également le dual de D.

Le dual possède de nombreuses propriétés utiles pour l'étude et la résolution d'un programme linéaire.

Proposition 1.2.10. *Théorème de dualité des programmes linéaires*

Soit P et D un programme linéaire et son dual comme définis précédemment, x et y des solutions réalisables de P et D, x^* et y^* des solutions optimales de P et D, alors :

$$b^\top y \leq b^\top y^* = c^\top x^* \leq c^\top x$$

Autrement dit, les optima du primal et du dual ont les mêmes valeurs d'objectif, et les solutions réalisables du dual fournissent des bornes sur l'optimum du primal (et réciproquement).

L'étude des solutions du dual est également intéressante pour l'interprétation qu'il est possible de faire des valeurs des variables duales. La k-ième variable duale de vecteur y est liée à la k-ième ligne du système d'inéquations défini par l'équation (1.2), qu'il est possible d'écrire sous la forme $\sum_{l=1}^{N} a_{kl}x_l \geq b_k$. La valeur de y_k correspond à la diminution de la valeur d'objectif du primal qui serait observée si on diminuait de 1 la valeur de b_k. Cette idée est au cœur des algorithmes de génération de colonnes.

Résolution de grands programmes linéaire par génération de colonnes

Comme évoqué précédemment, la résolution de programmes linéaire de taille raisonnable (c'est-à-dire, ayant un nombre polynomial de variables) peut être effectuée efficacement par un solveur de programmation linéaire, à l'aide de l'algorithme du simplexe par exemple. Néanmoins, on peut être amené à considérer des programmes linéaires faisant intervenir un trop grand nombre de variables (exponentiel) pour pouvoir être résolus explicitement par ces méthodes. En particulier, si le nombre de contraintes est faible, il est garanti qu'il existe une solution optimale pour laquelle le nombre de variables non-nulles est faible, appelée solution optimale de base. Les algorithmes de génération de colonnes s'appuient sur cette idée en choisissant un ensemble de variables restreint permettant d'arriver à une solution optimale.

Le principe est le suivant : une version restreinte du programme linéaire initial, faisant intervenir un sous-ensemble limité des variables initiales du problème $\mathcal{B}$ est considéré. Les éléments de ce sous-ensembles sont appelés *colonnes*. Les variables en dehors de l'ensemble des colonnes sont fixées à zéro. Ce nouveau sous-problème est appelé *Problème Maître Restreint* (RMP). La méthode est initialisée avec un ensemble de colonnes $\mathcal{B}_0$ bien choisi, de telle sorte que le RMP ait une solution réalisable. Ensuite, cette solution est améliorée itérativement en sélectionnant une nouvelle colonne susceptible d'améliorer notre solution à chaque étape avant de résoudre de nouveau le RMP. Cette nouvelle colonne est déterminée par la résolution d'un *problème de pricing* permettant de choisir la colonne la plus améliorante. La procédure s'arrête lorsqu'il n'existe plus de colonne améliorante. Dans ce cas, la solution optimale du RMP coïncide avec la solution optimale du programme linéaire initial. La méthode est résumée sur la figure 1.2.

Nous présentons dans le chapitre 5 un tel algorithme pour le calcul de la relaxation linéaire d'une variante de Bin-Packing, nous reviendrons plus en détail sur les aspects pratiques de la méthode.

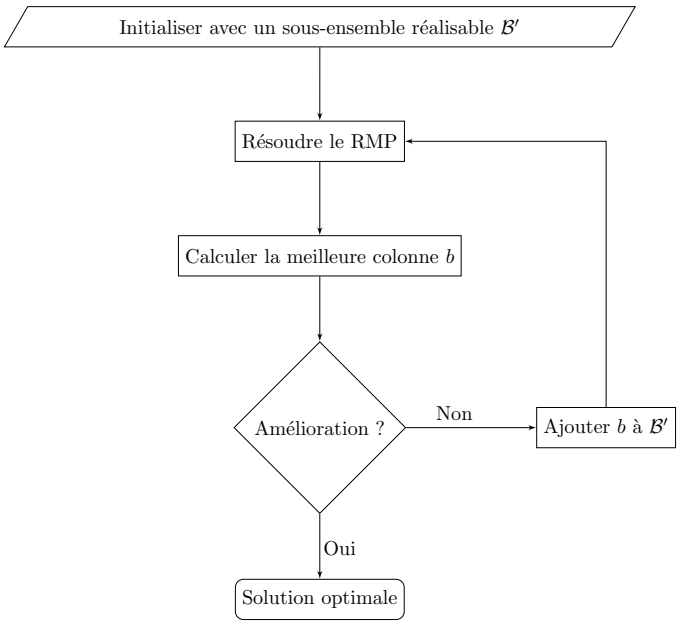


FIGURE 1.2 – Fonctionnement de la génération de colonnes

Programmation linéaire en nombres entiers

Définition 1.2.11. *Programme linéaire en nombres entiers (mixte)*

Un programme linéaire en nombres entiers mixte est un problème d'optimisation pouvant s'écrire de la manière suivante :

$$min. \quad c^\top x + d^\top z \tag{1.7}$$
$$s.\ t. \quad Ax + Bz \geq e \tag{1.8}$$
$$x \in \mathbb{R}^n_+,\ z \in \mathbb{N}^p \tag{1.9}$$

où $c \in \mathbb{R}^N$, $c \in \mathbb{R}^P$ et $b \in \mathbb{R}^M$ sont des vecteurs réels, et A et B sont des matrices réelles de dimension respectives $N \times M$ et $P \times M$.

Dans un programme linéaire en nombres entiers, un sous-ensemble ou la totalité des variables de décision sont entières. Les programmes linéaires en nombre entiers sont considérablement plus difficiles à résoudre que les programmes linéaires réels. Leur résolution est un problème NP-difficile.

En particulier, la Proposition 1.2.10 n'est plus que partiellement vérifiée, car on n'a plus forcément l'égalité entre les deux solutions optimales, mais seulement une inégalité.

On appelle *relaxation linéaire* d'un programme linéaire en nombres entiers P le programme linéaire P' où les variables entières de P ont été transformées en variables réelles. La solution optimale de la relaxation linéaire ne correspond pas nécessairement à l'optimum du programme linéaire en nombres entiers. Toutefois, l'ensemble des solutions du programme linéaire en nombres entiers est inclus dans l'ensemble des solutions de la relaxation linéaire puisqu'elles

respectent l'ensemble des contraintes. Cela implique que l'optimum de la relaxation est une borne inférieure pour le programme linéaire en nombres entiers. Cette propriété est très largement utilisée pour l'étude et la résolution de ces problèmes.

Les principaux solveurs de programmation linéaire en nombre entiers disponibles utilisent des méthodes de résolution de type branch-and-cut. Ces méthodes ont la particularité d'être des algorithmes anytime, fournissant au fur et à mesure de la résolution des solutions de plus en plus proche de l'optimum et s'arrêtant lorsque l'optimalité a été prouvée. Il est donc possible d'arrêter la résolution à tout moment avant la preuve d'optimalité pour obtenir une solution approchée. Dans ce cas, dans le cas de la minimisation, la meilleure solution obtenue est une borne supérieure de l'optimum. De plus, l'algorithme fournit également une borne inférieure, obtenue en résolvant par exemple une relaxation du problème. Ces bornes permettent d'avoir une garantie sur la proximité de la solution obtenue à l'optimum, et donc en définitive une garantie sur la qualité de la solution.

On appelle *gap absolu* la différence entre la valeur de la meilleure solution obtenue et la meilleure borne inférieure. Le *gap relatif* correspond au rapport du gap absolu sur la valeur de la meilleure solution obtenue et est généralement exprimé comme un pourcentage. Le gap relatif est une mesure de la proximité d'une solution à l'optimum. Un gap relatif faible signifie qu'il est certain que la solution est proche de l'optimum.

1.2.3 Optimisation multicritère

Classiquement, les problèmes étudiés en recherche opérationnelle font intervenir un seul objectif. Le plus souvent, il s'agit d'un profit à maximiser, de coûts à minimiser, ou d'un délai à minimiser. Néanmoins, pour de nombreuses situations réelles, il peut exister plusieurs critères à optimiser et l'arbitrage entre les différents critères peut être complexe. Pour aborder ces situations, il est possible d'étudier les interactions entre les critères pour identifier les solutions les plus pertinentes et accompagner la prise de décision.

Nous présentons dans cette section les concepts de base de l'optimisation multicritère. Cette partie s'appuie sur le livre *Multicriteria Scheduling : Theory, Models and Algorithms* de T'kindt et Billaut (2006).

Étant donnés un ensemble de solutions réalisables $\mathcal{S} \subset \mathbb{R}^n$, et une application $Z : \mathcal{S} \mapsto \mathbb{R}^K$ où f représente K critères d'optimisation. Le *problème d'optimisation multicritère* P consistant à trouver la solution $s \in \mathcal{S}$ optimisant les critères Z peut être défini de la manière suivante :

$$\text{min. } \{Z(s) | s \in \mathcal{S}\} \tag{1.10}$$

Néanmoins, contrairement à un problème d'optimisation mono-objectif, ici la minimisation ne porte plus sur des réels, mais sur des vecteurs de K réels. En l'absence d'ordre total naturel, la minimisation n'est pas proprement définie et il convient donc d'introduire de nouvelles définitions afin de caractériser les optima d'un tel problème. La définition proposée ici est très générale et concerne tout type de problème d'optimisation. Dans cette thèse, nous nous intéressons seulement au cas des programmes linéaires en nombres entiers multicritères, qui peuvent être écrits de la manière suivante :

$$\text{min.} \quad Z(x) = Cx + Dy \tag{1.11}$$
$$\text{s. t.} \quad Ax + By \geq e \tag{1.12}$$
$$x \in \mathbb{R}^Q, y \in \mathbb{Z}^P \tag{1.13}$$

où C et D sont respectivement des matrices réelles de dimension $K \times Q$ et $K \times P$ où K représente le nombre de critères (ou objectifs) , Q le nombre de variables réelles et P le nombre de variables entières. A et B sont également des matrices réelles de dimensions respectives $M \times Q$ et $M \times P$, et e un vecteur de $\mathbb{R}^M$, où M représente le nombre de contraintes.

En présence de plusieurs critères d'optimisation, la définition classique de l'optimalité n'est plus suffisante. En effet, l'optimum pour un objectif ne correspond pas nécessairement à l'optimum pour un autre, et dans ce cas le problème n'admet pas d'optimum au sens classique du terme. On s'intéressera donc plutôt à l'optimalité au sens de Pareto, définie ci-après.

Optimalité au sens de Pareto

Dans le cadre des problèmes d'optimisation classique, les solutions évoluent dans un espace noté $\mathcal{S}$ et l'image $Z(\mathcal{S})$ de cet espace par l'objectif est incluse dans la droite des réels, pour laquelle il existe un ordre naturel total clairement défini, et par conséquent la notion d'optimum est clairement définie.

Pour les problèmes d'optimisation multicritères, l'image de l'espace des solutions $\mathcal{S}$ par les objectifs définit l'*espace des critères* $Z(\mathcal{S})$, qui n'est plus inclus dans la droite, mais dans l'espace de vecteurs $\mathbb{R}^K$, dans lequel il n'existe pas de relation d'ordre totale naturelle.

Notons que dans le cas de la programmation linéaire en nombres entiers mixte, l'espace des solutions n'est pas connexe, et que par conséquent l'espace des critères n'est pas non plus nécessairement connexe.

On introduit la relation d'ordre partielle suivante dans $\mathbb{R}^K$:

$$\begin{aligned} \forall x, y \in \mathbb{R}^K,\ x \leq y &\iff \forall i \in \{1 \dots K\}\ x_i \leq y_i \\ x = y &\iff \forall i \in \{1 \dots K\}\ x_i = y_i \end{aligned}$$

Si $x \leq y$, dans le cas de la minimisation, on dit que x domine y. Cette définition est étendue par induction aux sous-ensembles de $\mathbb{R}^K$: $X \leq Y$ si et seulement si pour tout point $y \in Y$, il existe $x \in X$ tel que $x \leq y$.

En notant Z_i le i-ème critère, on définit l'optimalité au sens de Pareto de la manière suivante.

Définition 1.2.12. *Optimum de Pareto faible (T'kindt & Billaut, 2006)*

$x \in \mathcal{S}$ est un optimum de Pareto faible si et seulement s'il n'existe pas $y \in \mathcal{S}$ tel que $\forall i \in \{1 \dots K\},\ Z_i(y) < Z_i(x)$

En résumé, un optimum de Pareto faible est un point de l'espace des solutions tel qu'il n'existe pas d'autre point qui soit strictement meilleur pour chacun des critères. L'ensemble des optima de Pareto faibles est appelé *ensemble de Pareto*, et son image dans l'espace des

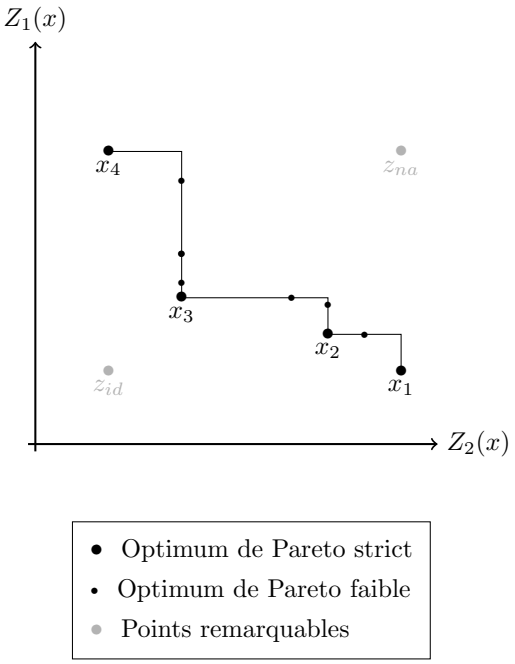


FIGURE 1.3 – Front de Pareto et points remarquables dans l'espace des critères

critères est appelé *front de Pareto*. Il existe dans la littérature de nombreux algorithmes permettant de calculer ou d'approcher le front de Pareto. Nous présentons dans le chapitre 3 la méthode ε-contrainte et ses aspects pratiques pour un problème bi-critère de planification de la maintenance.

En plus des optima de Pareto faible, on distingue également les optima de Pareto stricts, définis de la manière suivante.

Définition 1.2.13. *Optimum de Pareto strict (T'kindt & Billaut, 2006)*
$x \in \mathcal{S}$ est un optimum de Pareto strict si et seulement s'il n'existe pas $y \in \mathcal{S}$ tel que $\forall i \in \{1 \dots K\},\ Z_i(y) \leq Z_i(x)$ et qu'au moins une de ces inégalités est stricte.

L'ensemble des optima de Pareto stricts est en général plus intéressant que celui des optima de Pareto faibles, qui a peu d'intérêt du point de vue de l'aide à la décision.

On introduit également deux points remarquables :

Définition 1.2.14. *Point idéal (problème de minimisation) T'kindt et Billaut, 2006*
$z^{id} \in \mathbb{R}^K$ est le point idéal si $\forall i \in \{1 \dots K\},\ z_i^{id} = min_{x \in \mathcal{S}} Z_i(x)$

Définition 1.2.15. *Nadir (problème de minimisation) T'kindt et Billaut, 2006*
$z^{na} \in \mathbb{R}^K$ est le nadir si $\forall i \in \{1 \dots K\},\ z_i^{na} = max_{x \in \mathcal{S}} Z_i(x)$

L'ensemble de ces définitions est clarifié sur la Figure 1.3.

Par souci de lisibilité et pour aider la visualisation de l'allure des fronts de Pareto, nous relions les points successifs par des courbes en escaliers. Il convient toutefois de rappeler qu'en

pratique, le front de Pareto n'est pas nécessairement connexe et qu'un point quelconque situé sur la courbe ne correspond pas nécessairement à une solution réalisable.

Analyse du front de Pareto

En général, il est difficile et peu intéressant pour l'aide à la décision de calculer l'intégralité d'un front de Pareto, on se contente d'en calculer une approximation. Il est ainsi intéressant d'avoir des métriques permettant de comparer les différents fronts de Pareto obtenus en fonction des paramètres utilisés. Une grande variété d'indicateurs de qualité ont été considérés dans la littérature de l'optimisation multicritère. Les revues de Audet et al. (2021) et M. Li et Yao (2019) résument et classifient ces indicateurs en fonction de la qualité du front qu'ils permettent d'évaluer : convergence, dispersion, uniformité et cardinalité. Nous nous intéressons plus spécifiquement à trois critères : le temps de calcul total du front, le taux de solutions non-dominées et l'hypervolume.

Solutions non-dominées

Définition 1.2.16. *Une solution x d'un ensemble de Pareto faible X est non-dominée s'il n'existe pas de solution $y \in x$ telle que $y \leq x$.*

Il convient de remarquer qu'une solution non-dominée pour un ensemble de Pareto faible X donné n'est pas forcément strictement Pareto-optimale. En effet, X est un sous-ensemble des solutions Pareto-optimales faibles, et ne contient donc pas nécessairement tous les points Pareto-optimaux stricts. Dans ces conditions, un point de X peut être non-dominé sans être Pareto-optimal strict.

Une solution dominée présente peu d'intérêt pour l'aide à la décision, puisqu'on connaît une solution non-dominée qui est meilleure pour tous les critères. Le taux de solutions non-dominées est obtenu en calculant le rapport du nombre de solutions non-dominées divisé par le nombre total de solutions (Van Veldhuizen & Lamont, 2000). Un taux de solutions non-dominées important peut signifier qu'un grand nombre de solutions calculées sont redondantes, et que la méthode utilisée est peu efficace. Au contraire, un nombre de solutions non-dominées faible peut signifier que la méthode trouve efficacement des solutions pertinentes.

D'après M. Li et Yao (2019), le taux de solutions non-dominées est un bon indicateur pour évaluer la cardinalité d'un ensemble de Pareto.

Il est cependant nécessaire d'adosser cet indicateur à d'autres indicateurs pour évaluer la qualité des fronts calculés. Dans le cas d'un sous-échantillonnage du front de Pareto, le taux de solutions non-dominées est susceptible d'être élevé bien que l'ensemble de Pareto faible obtenu ne soit pas très représentatifs.

Hypervolume L'hypervolume est une autre métrique très largement utilisée pour l'étude de problèmes multi-critères. Cet indicateur est reconnu comme une bonne mesure de la qualité d'un front de Pareto, et possède de nombreuses propriétés utiles. En particulier, l'hypervolume est Pareto-compliant : si un ensemble A domine un ensemble B, alors l'hypervolume de A est supérieur à l'hypervolume B (Beume et al., 2009 ; Shang et al., 2021). Selon M. Li et Yao (2019), l'hypervolume est un bon indicateur de la convergence, de la dispersion et de

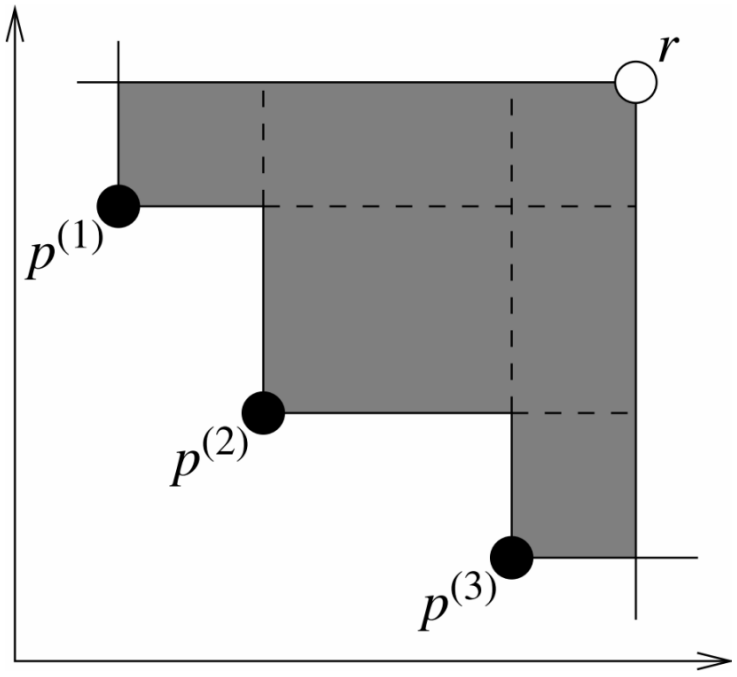


FIGURE 1.4 – Illustration de l'hypervolume. Image extraite de Fonseca et al. (2006) *An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator*

la cardinalité, et reflète également dans une moindre mesure l'uniformité d'un ensemble de Pareto faible.

Définition 1.2.17. *Hypervolume (Shang et al., 2021)*

Étant donné un ensemble de points $X \subset \mathbb{R}^K$ et un point de référence $r \in \mathbb{R}^K$, l'hypervolume est défini comme suit :

$$HV(X, r) = \mathcal{L}\left(\bigcup_{x \in X} \{y \in \mathbb{R}^K | r \leq y \leq x\}\right)$$

où $\mathcal{L}$ est la mesure de Lebesgue dans $\mathbb{R}^K$, c'est-à-dire $\mathcal{L}(X) = \int_X \mathbb{1}_{x \in X} dx$.

Concernant le choix du point de référence, les praticiens de l'optimisation multicritère recommandent habituellement de prendre un point "un peu moins bon que le nadir", sans en donner de définition précise. Les travaux récents dans le domaine remettent en cause la pertinence de choix. Il est toutefois indispensable de choisir le point de référence de telle sorte que les points extrêmes du front de Pareto soient inclus dans la mesure afin de conserver la propriété de Pareto-compliance (Ishibuchi et al., 2018).

Par souci de simplicité, dans cette thèse, nous utiliserons systématiquement le nadir comme point de référence. Intuitivement, calculer l'hypervolume consiste à tracer pour chaque point du front $x \in X$ le rectangle de diagonale $[r, x]$ dont les côtés sont parallèles aux axes, puis de calculer l'aire de l'union de tous ces rectangles, comme illustré sur la Figure 1.4.

L'hypervolume est souvent normalisé par l'aire du rectangle de construction similaire défini par le point idéal z^{id} et le point de référence r. Dans la suite du manuscrit, nous nous réfèrerons systématiquement à l'hypervolume normalisé.

Première partie

Planification tactique de la maintenance

La première partie de ce manuscrit concerne plus spécifiquement la prise de décision tactique dans le cadre du problème général défini en introduction.

Díaz-Madroñero et al. (2014) proposent une définition pour les modèles de planification tactique : il s'agit de calculer le meilleur usage des ressources disponibles sur un horizon de temps de l'ordre du mois ou de l'année en jouant sur les flux matériels, la gestion des stocks, les capacités de production, les quantités produites et la maintenance. La maintenance est donc une composante importante dans la prise de décision à l'échelle tactique.

L'objet de cette partie est de proposer un cadre large pour la modélisation et la résolution de problèmes de planification de la maintenance au niveau tactique avec prise en compte de la santé de l'équipement au moyen d'une approche basée sur la recherche opérationnelle.

Les travaux présentés dans cette partie ont fait l'objet d'une publication en conférence internationale (Foussard et al., 2021) et d'une publication soumise en revue.

Chapitre 2

Modélisation et propriétés structurelles

Sommaire

La division d'un problème en différents niveaux de décision et le traitement séparé de la décision tactique, sont très classiques dans certains champs d'application de la recherche opérationnelle. Elles permettent de définir des sous-problèmes plus simples, pouvant être modélisés et résolus séparément à l'aide des méthodes classiques de la littérature. Par exemple, en gestion de la chaîne logistique, ces techniques sont appliquées au niveau stratégique pour définir l'emplacement des installations, au niveau tactique pour dimensionner la production, au niveau opérationnel pour définir les plannings et les tournées de véhicules... Les différents sous-problèmes peuvent ensuite être recombinés dans une approche globale permettant de répondre à une situation industrielle complexe dans son ensemble. Ce dernier point est rarement abordé à travers le prisme de la recherche opérationnelle. Les approches provenant de domaines tels que l'automatisme sont généralement préférées.

Du côté de la planification de la maintenance, la séparation et l'identification de problèmes correspondant à chaque niveau de décision existent dans la littérature (Marquez & Gupta, 2006), bien que l'essentiel des travaux récents à notre connaissance soient des approches globales provenant de champs disciplinaires tels que le pronostic. Les approches concernant un seul niveau de décision s'intéressent principalement aux niveaux stratégiques (Murthy et al., 2002) et opérationnel (Froger et al., 2016 ; Malik, 1979), mais il existe assez peu de travaux concernant la seule planification tactique de la maintenance en particulier dans le domaine de la recherche opérationnelle. Des approches existent dans des contextes très spécifiques tels que le ferroviaire (Baldi et al., 2016 ; D'Ariano et al., 2019) ou l'aéronautique (Keysan et al., 2010 ; M. Masmoudi & Haït, 2010). La littérature de la planification de la production s'est également emparée du sujet, en raison de l'interconnexion entre la production et la maintenance des équipements, et des enjeux industriels concernant la maintenance prédictive. On peut ainsi trouver de nombreux travaux récents concernant la planification conjointe de la production et de la maintenance au niveau tactique. Fitouhi et Nourelfath (2014) étudient l'optimisation conjointe de la maintenance conditionnelle et de la production sur un système multi-composant. Jafari et Makis (2015) s'intéressent à un problème de lot-sizing sujet à des dégradations modélisées par un procédé de Markov dépendant de l'âge de l'équipement et de l'information provenant de capteurs. Peng et van Houtum (2016) étudient un problème de lot-sizing stochastique intégrant la maintenance conditionnelle en considérant un espace d'états continus pour la dégradation et étudient l'impact à long terme de la dégradation. Shamsaei et Van Vyve (2017) se sont intéressés à l'intégration de plannings de maintenance cyclique et conditionnelle à un problème de lot-sizing. Gao et al. (2020) étudient un problème de planification conjointe de la maintenance et de la production, où deux niveaux de pannes sont considérés et modélisés par un procédé stochastique . Lemoine et Castanier (2021) proposent une approche intégrée pour la planification tactique de la production et de la maintenance prédictive basée sur un indicateur permettant de modéliser l'impact de la dégradation du système sur sa performance.

L'approche proposée dans cette partie se démarque des travaux évoqués ci-dessus puisqu'elle vise à appliquer des méthodes classiques de recherche opérationnelle à la planification de la maintenance seule au niveau tactique. À notre connaissance, ce type d'approche n'existe pas dans la littérature.

2.1 Définition du problème

Le problème consiste à trouver un planning de maintenance pour une unique machine à plusieurs composants $\mathcal{G} = \{1 \dots G\}$ sur un horizon de temps divisé en T périodes $\mathcal{T} = \{1 \dots T\}$.

Ce planning doit minimiser des objectifs de coûts économiques et de consommation de ressources, décrits dans les sections suivantes.

2.1.1 Environnement machine

Chaque composant $g \in \mathcal{G}$ de la machine possède un niveau de santé compris entre 0 et 100. Le niveau de santé au début de la période $t \in \mathcal{T}$ est noté H_{gt}. On note également $H_{g,0}$ la santé initiale de chaque composant $g \in \mathcal{G}$.

A chaque période de temps, la machine dispose d'une capacité de production théorique $q_t \in \mathbb{R}_+$. Par défaut, la machine produit au maximum de sa capacité. L'exploitation cause une usure dépendante de la quantité produite sur chacun des composants à chaque période. On note w_g la diminution de santé par unité de demande satisfaite subie par le composant g.

Un ensemble d'activités de maintenance $m \in \mathcal{M} = \{1 \dots M\}$ permet de régénérer la santé de chaque composant g d'un montant fixe r_{gm}. On distinguera en particulier le cas de la maintenance parfaite, régénérant la santé à 100 pour les composants concernés, et celui de la maintenance ciblée, concernant un unique composant. Dans les faits, lorsqu'un équipement subit une maintenance, cela induit la plupart du temps une indisponibilité pour l'exploitation. Dans notre problème, à l'échelle tactique, cela se traduit par une perte d'une fraction de la capacité de production $p_m \in [0, 1]$ sur la période considérée. On considère, sauf mention explicite du contraire, que chaque maintenance peut être effectuée autant de fois que nécessaire sur l'horizon de temps, mais pas plusieurs fois pendant la même période. Il est toutefois possible d'avoir plusieurs maintenances sur une même période à condition qu'elles soient différentes.

On note P le plan de maintenance, et P_t l'ensemble des maintenances planifiées au cours de la période t. Pour un plan donné, on peut calculer la valeur de l'usure subie W_{gt}, proportionnelle à la capacité effective (i.e. non perdue pour cause de maintenance) de production, à chaque période :

$$W_{gt} = q_t w_g (1 - \sum_{m \in P_t} p_m) \tag{2.1}$$

La régénération R_{gt} reçue à chaque période est quant à elle égale à la régénération apportée par chacune des maintenances planifiées pendant la période :

$$R_{gt} = \sum_{m \in P_t} r_{gm} \tag{2.2}$$

2.1.2 Contraintes

Conservation du flot Une première contrainte concerne la conservation de la santé et correspond à une contrainte classique de conservation du flot. Cette contrainte impose qu'au début d'une période t et pour tout composant g, la santé H_{gt} soit égale à la santé de ce

composant à la période précédente H_{gt-1}, à laquelle on retire la dégradation subie pendant la période et ajoute la régénération reçue en cas de maintenance.

Il y a toutefois un cas particulier de non-conservation du flot, lorsque la régénération entraînerait un dépassement du plafond de santé 100. Dans ce cas, la maintenance reste possible, mais la nouvelle valeur de la santé est écrêtée à 100.

Formellement, la contrainte s'écrit donc :

$$\forall g \in \mathcal{G},\ \forall t \in \mathcal{T},\ H_{gt} = min\left\{100, H_{gt-1} - W_{gt-1} + R_{gt-1}\right\} \tag{2.3}$$

Capacité de production Une autre contrainte porte sur la compatibilité des maintenances en raison de la perte de capacité qu'elles entraînent : on suppose que les maintenances sont mutuellement exclusives, c'est-à-dire que lorsque deux maintenances sont planifiées sur la même période, les pertes de capacité de production s'additionnent. La perte de capacité totale ne peut quant à elle pas excéder la capacité disponible. Ce qui se traduit formellement de la manière suivante :

$$\forall t \in \mathcal{T},\ \sum_{m \in P_t} p_m \leq 1 \tag{2.4}$$

2.1.3 Critères d'optimisation

Minimisation des coûts économiques Les coûts économiques sont tous liés à la maintenance de manière directe ou indirecte, et se décomposent en trois termes.

$$COUTS = CF + CM + CO \tag{2.5}$$

Le premier terme CF correspond aux coûts fixes classique des problèmes de lot-sizing. Un coût fixe s est chargé dès lors qu'au moins une maintenance est planifiée sur une période.

$$CF = \sum_{t \in \mathcal{T}} s \mathbb{1}_{P_t \neq \emptyset} \tag{2.6}$$

Le terme CM correspond aux coûts individuels de maintenance. Chaque maintenance de type m planifiée induit un coût c_m.

$$CM = \sum_{t \in \mathcal{T}} \sum_{m \in P_t} c_m \tag{2.7}$$

On considère en plus de cela un coût d'opportunité représentant les bénéfices manqués liés à la perte de capacité de production en raison de la maintenance. On note ρ le rendement économique d'une unité de production, ce coût d'opportunité s'exprime de la manière suivante :

$$CO = \rho \sum_{t \in \mathcal{T}} \sum_{m \in P_t} q_t p_m \tag{2.8}$$

Minimisation de la consommation de ressource Nous introduisons un second objectif de minimisation de la consommation de ressources, dépendant de la santé de l'équipement : celui-ci est décomposé en deux termes.

$$RESS = CRM + CRE \tag{2.9}$$

Le premier terme, CRM correspond à la consommation de ressources liée aux opérations de maintenance planifiées sur l'équipement. Chaque opération de maintenance m induit une consommation de ressource fixe α_m. On a alors l'expression suivante :

$$CRM = \sum_{t \in \mathcal{T}} \sum_{m \in P_t} \alpha_m \tag{2.10}$$

Le second terme, CRE correspond à la consommation de ressources liée à l'exploitation de l'équipement usé. Nous proposons une modélisation comme une fonction linéaire de la santé manquante de l'équipement et de la capacité effective de production :

$$CRE = \sum_{t=1}^{T} \sum_{g=1}^{G} \beta_g (100 - H_{gt})(q_t - \sum_{m \in P_t} p_m q_t) \tag{2.11}$$

où $\beta_g \in \mathbb{R}_+$ est un coefficient de proportionnalité.

Nous revenons plus en détail sur cet objectif dans le chapitre 3.

2.1.4 Exemple

Afin de clarifier le problème, un exemple est proposé à titre d'illustration avec $T = 5$ périodes pour une machine mono-composant, avec les valeurs suivantes pour les paramètres :

— taille du problème : $T = 5$, $G = 1$, $M = 1$

— coûts et profits : $s = 10$, $c_1 = 1$, $\rho = 0.4$, $p_1 = 50\%$, $\alpha_1 = 50$, $\beta_1 = 0.1$

— environnement machine : $H_{1,0} = 90$, $r_{11} = 50$, $w_1 = 0.4$

On considère les capacités de production suivantes :

t	1	2	3	4	5
q_t	50	50	100	50	50

Deux solutions sont considérées. Dans le cas du plan de maintenance P_1 (resp. P_2), une unique maintenance est réalisée à l'instant $t = 3$ (resp. $t = 4$). Le calcul détaillé des différents termes de l'expression des coûts et de la santé sont indiqué dans les tables 2.1 et 2.2. Plus précisément, la Table 2.1 correspond aux caractéristiques de l'environnement machine : santé au début de la période ; capacité de production effective, obtenue par soustraction de la capacité de production perdue $p_1 q_1$ à la capacité de production théorique q_1 ; l'usure et la régénération subies par l'unique composant au cours de la période de temps. Dans la Table 2.2, ce sont les valeurs par période du coût économique et des deux termes de l'objectif de consommation de ressources qui sont représentées. Les calculs sont détaillés davantage par la suite.

L'impact du plan de maintenance sur la capacité de production effective est représenté sur la Figure 2.1. La capacité de production théorique est représentée par la hauteur des rectangles et la hauteur des hachures correspond à la fraction de cette capacité de production

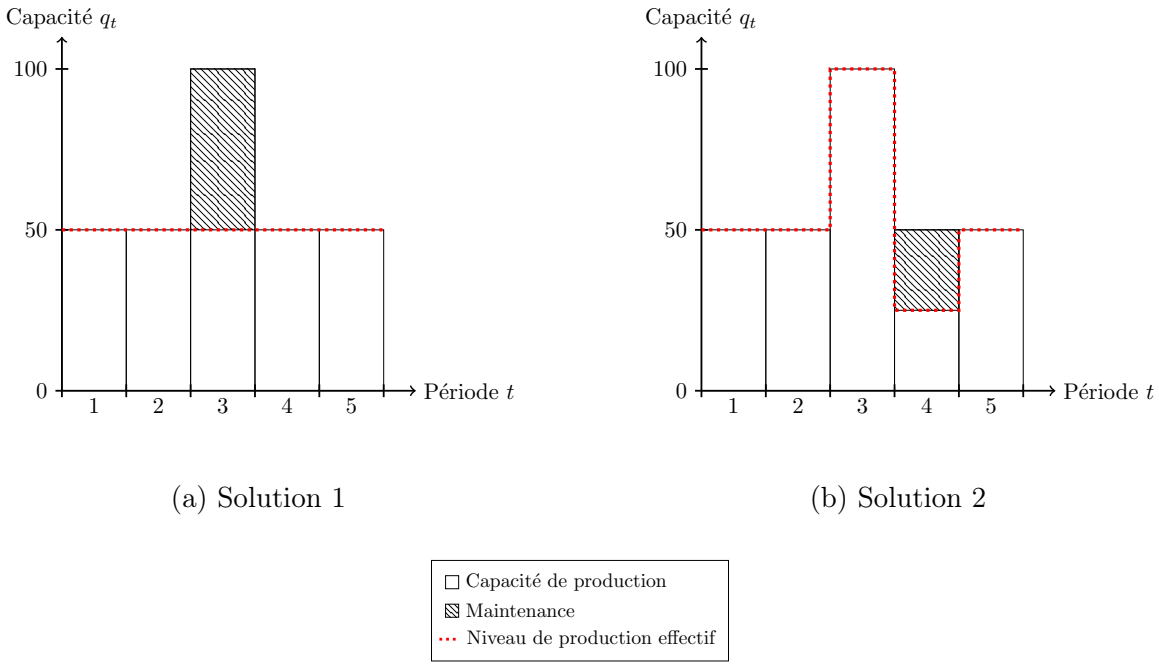
	Solution 1					Solution 2				
Période t	1	2	3	4	5	1	2	3	4	5
Niveau de santé $H_{1,t}$	90	70	50	80	60	90	70	50	10	50
Capacité effective	50	50	50	50	50	50	50	100	25	50
Usure	20	20	20	20	20	20	20	40	10	20
Régénération	0	0	50	0	0	0	0	0	50	0

TABLE 2.1 – Détails de calcul pour l'exemple : environnement machine

	Solution 1					Solution 2				
Période t	1	2	3	4	5	1	2	3	4	5
Coût économique	0	0	31	0	0	0	0	0	21	0
$CRM(t)$	0	0	50	0	0	0	0	0	50	0
$CRE(t)$	50	150	250	100	200	50	150	500	225	250

TABLE 2.2 – Détails de calcul pour l'exemple : coûts et consommation de ressources

perdue pour cause de maintenance. La capacité de production effective est représentée par la ligne rouge en pointillés. L'évolution du niveau de santé pour les deux solutions est représentée sur la Figure 2.2. Au cours de la première période $t = 1$, aucune maintenance n'est planifiée



(a) Solution 1

(b) Solution 2

FIGURE 2.1 – Deux solutions pour l'instance de l'exemple.

dans les deux solutions. La capacité de production théorique étant de $q_1 = 50$ et le taux d'usure $w_1 = 0.4$, la dégradation totale est $q_1 w_1 = 20$ dans les deux cas. La valeur de la santé

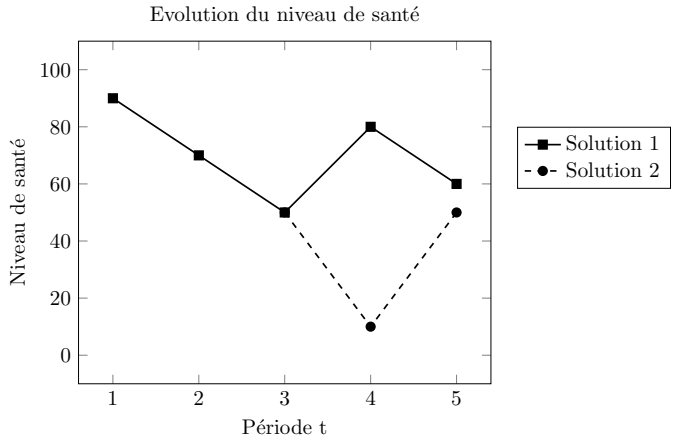


FIGURE 2.2 – Évolution du niveau de santé pour les deux solutions de l'exemple.

passe alors de 90 à 70. Le fonctionnement est similaire pour la deuxième période. La période $t = 3$ de la Solution 1 permet d'illustrer le comportement de la contrainte de conservation du flot en présence de la maintenance. On a d'une part la régénération de la santé de $r_{11} = 50$, et d'autre part l'usure causé par la capacité de production effective : la maintenance cause une perte $p_1 = 50\%$ de la capacité de production. Cette production restante cause une usure égale $q_3(1 - p_1)w_1 = 20$. Au total, la santé est augmentée de 30, passant de 50 à 80. Notons que si la valeur de la régénération était plutôt $r'_{11} = 100$, effectuer la maintenance reste possible malgré le plafond de santé, toutefois, la santé de l'équipement serait écrêtée à 100 plutôt que d'atteindre la valeur théorique de 130. Chaque solution contient exactement une maintenance du même type, représentant un coût direct de $CF + CM = 1 \cdot s + 1 \cdot c = 11$. Du côté du coût d'opportunité, le rendement d'une unité de production est fixé à $\rho = 0.4$. Dans le cas de la Solution 1, la moitié de la capacité de la période $t = 3$ est perdue, causant un coût d'opportunité égal à $\rho q_3 p_1 = 20$. Pour la Solution 2, c'est la capacité de la période $t = 4$ qui est diminuée de moitié, soit un coût d'opportunité $\rho q_4 p_1 = 10$. Le coût économique total de la Solution 1 est donc 31, contre 21 pour la Solution 2, qui est donc plus avantageuse sur le plan économique.

En ce qui concerne la consommation de ressources, le terme lié à la maintenance est identique pour les deux solutions, puisqu'elles contiennent toutes les deux exactement une maintenance, soit $CRM = 50$.

Le calcul du terme d'exploitation CRE est illustré sur la période $t = 3$. Pour la première solution, la maintenance est réalisée au cours de la période, avec une perte de capacité de production $p_m = 50\%$, et avec un niveau de santé au début de la période $H_{1,3} = 50$. Le coût en ressources pour cette solution vaut $\beta_1(q_3 - q_3 p_1)(100 - H_{13}) = 250$. On a un calcul similaire dans le cas de la Solution 2, mais cette fois-ci aucune maintenance n'est planifiée, ce qui signifie que toute la capacité de production est utilisée. Le coût en ressource vaut alors $\beta_1 q_3(100 - H_{13}) = 500$. Sur l'ensemble de l'horizon de temps, on obtient une consommation de ressource totale de 750 pour la Solution 1, contre 1175 pour la Solution 2. Notons qu'ici nous traitons du niveau tactique de décision, mais cette décision peut avoir des

répercussions au niveau opérationnel. Par exemple, si le niveau de santé est bas, comme c'est le cas pour la période $t = 4$ dans la Solution 2, supposons que la maintenance est planifiée tard sur la période, alors cela signifierait que la santé serait temporairement négative, ce qui est contradictoire. Au niveau opérationnel, il sera nécessaire de prendre en compte cette contrainte dans l'ordonnancement, mais au niveau tactique on se contentera de définir des volumes de production et de maintenance sans se préoccuper de leur ordonnancement.

2.2 Lot-sizing

Dans cette section, les liens entre notre problème de planification tactique de maintenance et les problèmes de lot-sizing, habituellement rencontrés en planification de la production, sont identifiés et approfondis. Dans un premier temps, nous nous intéressons au positionnement bibliographique des contraintes et caractéristiques du problème, puis nous passons brièvement en revue les techniques classiques pour la modélisation et la résolution de ce type de problèmes.

2.2.1 Liens avec le lot-sizing

En planification de production, les problèmes de lot-sizing consistent à déterminer à moyen terme quand et combien produire afin de minimiser les coûts de démarrage de l'équipement, de production et de stockage (Karimi et al., 2003). Il s'agit donc de déterminer des tailles de lots optimales de sorte qu'il n'y a pas besoin de mettre en marche trop souvent l'équipement, ni de stocker des quantités de produit importantes sur des durées étendues.

Le problème de lot-sizing standard est la version sans capacité à un seul item (SILSP) telle que définie par Wagner et Whitin (1958). Les principales caractéristiques de ce problème sont la présence d'un objectif économique faisant intervenir les coûts de démarrage, de production et de stockage, et la contrainte de conservation du flot.

La contrainte de conservation du flot impose que le niveau de stockage à l'instant $t + 1$ est égal au niveau de stockage à l'instant t, auquel on ajoute la quantité produite à l'instant t et soustrait la quantité de demande satisfaite.

Liens avec le SILSP Le problème étudié ici est un problème de planification de la maintenance, et non de la production. Toutefois, il s'agit de deux problématiques intrinsèquement liées : la maintenance prédictive permet d'allonger la durée d'exploitation de l'équipement, et par conséquent la quantité produite. Il s'agit donc d'une forme de production différée. Dans ce cadre, il devient alors possible de traiter le niveau de santé comme un potentiel de production disponible et le problème de planification de la maintenance consiste alors à définir des volumes de production potentielle. On peut alors appliquer des raisonnements typiques de la planification de la production.

Les analogies entre les concepts de notre problème de planification de maintenance et ceux du SILSP sont listés ci-après :

— Régénération - Quantité produite
— Niveau de santé - Niveau d'inventaire
— Usure - Demande
— Coûts de maintenance - Coûts de production

Au niveau de l'objectif, on retrouve les mêmes structures. Le coût fixe s est équivalent au coût de démarrage du lot-sizing et la somme du coût de maintenance et du coût d'opportunité est similaire au coût de production. Toutefois, il existe une différence notable. Le niveau de santé n'induit pas de coût, contrairement au stockage en lot-sizing.

La régénération, le niveau de santé et l'usure sont liés par une loi de conservation du flot similaire à celle qui relie production, niveau d'inventaire et demande en lot-sizing. Le terme d'usure présente cependant une différence, puisqu'il dépend de la capacité de production perdue pour cause de maintenance, et donc finalement de la décision de maintenance. Il n'existe pas à notre connaissance dans la littérature du lot-sizing de terme de demande qui soit dépendant de la taille de lots produits.

Le problème que nous étudions fait intervenir d'autres contraintes et caractéristiques que celle du SILSP. Certaines de ces caractéristiques ont un équivalent dans la littérature du lot-sizing.

Régénérations fixes - Taille de lots prédéfinies Une des principales originalités de notre problème par rapport au SILSP est que les montants de régénération, analogue à la taille des lots, sont une donnée du problème : chaque maintenance permet de régénérer un montant fixe. Une telle contrainte n'existe pas à notre connaissance dans la littérature du lot-sizing. Toutefois, elle est apparentée à d'autres contraintes précédemment étudiées.

Dans le problème de lot-sizing avec taille de lots constante (SILSP-SW), les quantités produites à chaque période doivent être un multiple d'une taille standard fixée en entrée du problème. Cette contrainte permet notamment de modéliser les cas où les quantités produites doivent ensuite être transportées dans des containers ou en camion, ce qui impose des restrictions sur les tailles de lots. La contrainte a été introduite par Lippman (1969) pour modéliser les transports en camion et a ensuite fait l'objet de plusieurs travaux. Van Vyve (2007) étudie le cas du SILSP-SW avec backlogging. Akbalik et Rapine (2012) s'intéressent au cas où une capacité de production est considérée en plus des tailles de lots constantes. Akbalik et Rapine (2013) étudient les différentes variantes du cas où la taille des lots dépend du temps. Le SILSP-SW est généralisé par les problèmes de lot-sizing avec structure de coûts de production concave par morceaux (Brahimi et al., 2017).

Cette contrainte sur la taille de lots peut être également comparée aux contraintes sur la capacité de production, très largement étudiées en lot-sizing. Les travaux concernant cette variante ont fait l'objet de plusieurs revues de littérature par Karimi et al. (2003), par Brahimi et al. (2006) et par Brahimi et al. (2017).

Notre contrainte se distingue toutefois puisqu'elle permet de choisir parmi plusieurs tailles de lots possibles, ce qui est une généralisation des deux types de contraintes classiques cités précédemment.

Multi-composant - Multi-item L'aspect multi-composant peut-être vu comme un cas particulier du lot-sizing multi-item. La particularité dans notre cas est que les usures par composant, analogue à la demande par item, ne sont pas totalement indépendantes puisqu'elles sont toutes proportionnelles à la capacité de production de la période considérée. Les principales revues de littérature intégrant le cas du lot-sizing multi-item ont été proposées par Barany et al. (1984) et Karimi et al. (2003). Il n'existe pas à notre connaissance de revue de littérature plus récente concernant le problème de lot-sizing multi-item.

Santé maximale - Capacité d'inventaire Une contrainte notable de notre problème est le plafond de santé fixé à 100. Cette contrainte est comparable aux contraintes de capacité d'inventaire, qui ont fait l'objet de nombreux travaux, y compris récemment, en lot-sizing. Cette contrainte a été introduite par Love (1973) pour le problème de lot-sizing standard avec coûts concaves. Dans cette première approche, les auteurs considèrent un niveau minimum et un niveau maximum de stock simultanément . Atamtürk et Küçükyavuz (2008) citent le domaine de la chaîne d'approvisionnement bio-pharmaceutique et le stockage de données comme exemple où l'absence de capacité d'inventaire n'est pas une hypothèse réaliste, et s'intéressent plus particulièrement au cas des coûts fixes . Les travaux plus récents s'intéressent à la combinaison des capacités d'inventaires appliqués à certaines variantes du lot-sizing. Hwang et Van Den Heuvel (2012) s'intéressent au cas du backlogging en présence de la capacité d'inventaire. Akbalik et al. (2015a) étudient le problème de lot-sizing multi-item avec capacité d'inventaire. Dans un autre article publié la même année, les mêmes auteurs s'intéressent au même problème, mais en présence également de contrainte de capacité de production (Akbalik et al., 2015b). Phouratsamay et al. (2018) étudient un problème de lot-sizing bi-niveau avec contraintes de Witt (2019) s'intéresse au cas multi-niveau avec contrainte de capacité de production et d'inventaire. Jing et Mu (2020) considèrent le cas des produits périssables avec substitution de demande en présence de la contrainte de capacité d'inventaire.

2.2.2 Positionnement méthodologique

En raison des forts liens entre nos contraintes et celles des problèmes de lot-sizing et variantes plus classiques, il est intéressant d'étudier les méthodologies les plus utilisées pour la résolution du lot-sizing.

Plusieurs revues de littérature existent à ce sujet, elles sont citées de la plus ancienne à la plus récente. Karimi et al. (2003) recensent les méthodes existantes pour le lot-sizing avec et sans capacité de production. Brahimi et al. (2006) proposent une revue très complète des problèmes de lot-sizing mono-item. Cette revue a été mise à jour par Brahimi et al. (2017). Jans et Degraeve (2008) proposent une vue d'ensemble des techniques de modélisation pour les problèmes de lot-sizing industriels et leurs extensions. Enfin, Glock et al. (2014) proposent une revue systématique des revues de littérature concernant le domaine du lot-sizing, et proposent une synthèse des différents courants de recherche existants.

En pratique, la programmation dynamique est très utilisée pour résoudre efficacement les variantes les plus simples du lot-sizing. Toutefois, l'ajout de contraintes supplémentaires comme des capacités de production rend rapidement ces problèmes NP-difficiles au sens fort, ce qui rend inapplicables ce type de méthodes. Nous rencontrons également cette difficulté dans le cadre de notre problème, qui est également NP-difficile au sens fort comme démontré dans la section 2.3.1.

Les principales méthodes utilisées dans ces cas-ci sont la programmation linéaire en nombres entiers pour la résolution exacte et des méthodes heuristiques. Cela a motivé le développement de méthodes basées sur la programmation linéaire en nombres entiers pour la modélisation et la résolution de notre problème.

Dans leur revue sur la modélisation des problèmes de planification tactique de la production, Díaz-Madroñero et al. (2014) confirment l'omniprésence des modèles de programmation linéaire en nombre entier pour ce type de problèmes.

2.3 Modélisation et étude théorique

Dans cette section, nous nous établissons les principaux résultats théoriques concernant le problème : NP-complétude et modélisation en programmation linéaire en nombre entiers. La résolution pratique du problème et l'approche multi-critère font l'objet du chapitre 3.

2.3.1 Complexité

Nous nous intéressons dans un premier temps à la complexité du problème. Deux variantes du problème sont considérées : $\mathcal{P}_{mult}$, où chaque maintenance peut être planifiée autant de fois que nécessaire, et $\mathcal{P}_{single}$, où une opération de maintenance peut avoir lieu au plus une seule fois. Ainsi, dans $\mathcal{P}_{mult}$, les éléments de $\mathcal{M}$ correspondent à des types de maintenances, sans limitation sur le nombre d'utilisations, tandis que dans $\mathcal{P}_{single}$, cela correspond à des opérations de maintenance individuelles, ne pouvant être planifiées qu'une seule fois. On note respectivement $\mathcal{D}_{mult}$ et $\mathcal{D}_{single}$ les problèmes de décision correspondants. On note C la borne supérieure sur la valeur de l'objectif économique dans ces problèmes de décision. Tout d'abord, nous prouvons que $\mathcal{D}_{mult}$ est NP-complet au sens fort. Puis, nous montrons que $\mathcal{D}_{mult}$ peut être réduit dans $\mathcal{D}_{single}$, ce qui implique que $\mathcal{D}_{single}$ est également NP-complet au sens fort. Enfin, nous prouvons que même dans le cas à un seul composant $G = 1$, le problème $\mathcal{D}_{single}$ reste NP-complet au sens fort.

Dans la suite, on ne considérera que l'objectif de minimisation du coût économique. L'objectif de ressources n'est pas considéré. Toutefois, les résultats de NP-complétude établis ci-après restent valables puisque l'ajout de ce second objectif augmente la complexité du problème.

NP-complétude des problèmes $\mathcal{D}_{mult}$ et $\mathcal{D}_{single}$

Toutes les réductions présentées dans ce chapitre consistent à réduire 3-PARTITION dans le problème considéré. Le problème 3-PARTITION et les notations associées sont introduites ci-après.

Définition 2.3.1. *3-PARTITION (Garey & Johnson, 1990)*

Instance *: Un ensemble fini d'items A, $|A| = 3l$, $l \in \mathbb{N}$, une borne $K \in \mathbb{N}$ et une taille $s(a) \in \mathbb{N}$ pour tout $a \in A$ tels que $\forall a \in A, K/4 < s(a) < K/2$ et $\sum_{a \in A} s(a) = lK$.*

Question *: Existe-t-il une partition de A en l ensembles disjoints $A_1 \dots A_l$ tels que pour tout $1 \leq i \leq l$, $\sum_{a \in A_i} s(a) = K$?*

NP-complétude de $\mathcal{D}_{mult}$

Proposition 2.3.2. *$\mathcal{D}_{mult}$ est NP-complet au sens fort.*

Démonstration. Soit I, une instance de 3-PARTITION. Notons f, l'application transformant une instance de 3-PARTITION en une instance de $\mathcal{D}_{mult}$ de la manière suivante. L'équipement est constitué de $G = 3l$ composants, et on considère $M = 3l$ maintenances, de telle sorte qu'il existe deux bijections $\gamma : A \to \mathcal{G}$ et $\mu : A \to \mathcal{M}$ entre les A et les ensembles respectifs de composants et de maintenances. L'horizon de temps est fixé à $T = l + 1$. Pour tout item $a \in A$,

les activités de maintenance $\mu(a)$ causent une perte de capacité de production $p_{\mu(a)} = s(a)/K$, et sont parfaites et ciblées sur le composant $\gamma(a)$. Soit :

$$\forall (a_1, a_2) \in A^2,\ r_{\gamma(a_1)\mu(a_2)} = \begin{cases} 100 & \text{si } a_1 = a_2\ , \\ 0 & \text{sinon} \end{cases}$$

Les coûts liés directement à la maintenance sont nuls $s = 0$, $\forall a \in A,\ c_{\mu(a)} = 0$. Les capacités de productions sont fixées $q_t = 0$ pour les l premières périodes, et $q_{l+1} = 1$. On considère qu'une unité de production a un rendement $\rho = 1$, et consomme toute la santé de tous les composants $w_{\gamma(a)} = 100$. La santé initiale est nulle pour tous les composants $\forall a \in A,\ H_{\gamma(a),0} = 0$. Le problème de décision à résoudre s'énonce de la manière suivante : " Existe-t-il un plan de maintenance tel que le coût soit égal à 0 ? ". Le but est donc de régénérer tous les composants lors des l premières périodes pour permettre une production totale pendant la dernière période et éviter tout coût d'opportunité lié à la maintenance afin d'avoir un objectif égal à 0. Notons $Y_{\mathcal{D}_{mult}}$ (resp. Y_{3P}) l'ensemble des instances "oui" de $\mathcal{D}_{mult}$ (resp. 3-PARTITION), montrons que $I \in Y_{\mathcal{D}_{mult}} \iff f(I) \in Y_{3P}$:

$\Rightarrow$ Si $I \in Y_{3P}$, on note A_{ij} avec $i \in \{1 \dots l\}$ et $j \in \{1, 2, 3\}$ le j-ième item du i-ième sous ensemble de la partition. Alors le plan de maintenance $P = (P_t)_{t \in \{1 \dots l+1\}}$ où $\forall t \in \{1 \dots l\},\ P_t = \{\mu(A_{t1}), \mu(A_{t2}), \mu(A_{t3})\}$ et $P_{l+1} = \emptyset$ est réalisable avec coût 0. D'où $f(I) \in Y_{\mathcal{D}_{mult}}$. Ce planning est représenté sur la Figure 2.3. La perte totale de production $\sum_{m \in P_t} p_m$ est représenté sur l'axe des ordonnées et les périodes sont représentées sur l'axe des abscisses. Les rectangles gris représentent pour chaque maintenance la perte de production associée. Les pointillés rouges représentent la capacité de production restante.

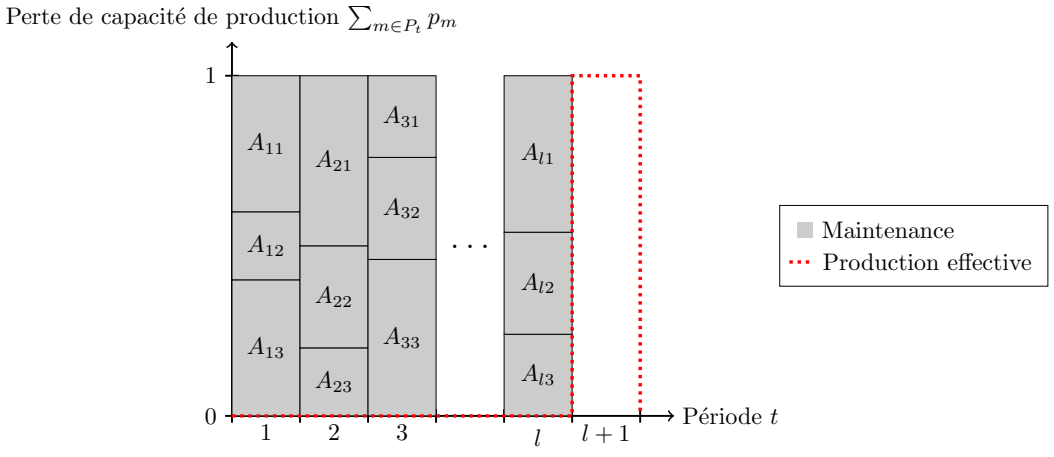


FIGURE 2.3 – Réduction de 3-PARTITION dans $\mathcal{D}_{mult}$

$\Leftarrow$ On suppose à présent que I est une instance de 3-PARTITION telle que $f(I) \in Y_{\mathcal{D}_{mult}}$, c'est-à-dire qu'il existe un plan de maintenance dont le coût est 0. Par hypothèse :

$$COUTS = CO = \sum_{t=1}^{l+1} \rho q_t \sum_{\mu(a)\in P_t} p_{\mu(a)} = \sum_{\mu(a)\in P_{l+1}} p_{\mu(a)} = 0$$

Or, puisque I est une instance de 3-PARTITION, on a $p_{\mu(a)} = \frac{s(a)}{K} > \frac{K/4}{K} = 0,25 > 0$. Un coût égal à 0 ne peut être obtenue que si $P_{l+1} = \emptyset$. Cela implique que la production doit être totale sur la dernière période. Sous cette hypothèse, la dégradation subie par chaque composant $\gamma(a)$ sur cette dernière période est égale à $w_{\gamma(a)}q_{l+1} = 100$. Pour que la solution soit réalisable, il faut donc que la santé de chaque composant au début de la période $l+1$ soit égale à 100. Puisque $H_{\gamma(a),0} = 0$ pour tous les composants, chaque maintenance doit être effectuée une fois au cours des l premières périodes. Par hypothèse, $\sum_{a\in A} s(a) = 100l$ et $0.25 < p_{\mu(a)} < 0.5$. Comme la perte de capacité de production ne peut pas excéder 100% : $\forall t \in \mathcal{T}, \sum_{m\in P_t} p_m \leq 1$, exactement trois maintenances sont planifiées dans chacune des l premières périodes, et toutes les maintenances doivent être planifiées. Cela correspond à une instance réalisable de 3-PARTITION, d'où $I \in Y_{3P}$.

□

Par conséquent, le problème $\mathcal{P}_{mult}$ est NP-difficile au sens fort.

Réduction de $\mathcal{D}_{mult}$ dans $\mathcal{D}_{single}$

Proposition 2.3.3. *$\mathcal{D}_{mult}$ est réductible dans $\mathcal{D}_{single}$.*

Toute instance de $\mathcal{D}_{mult}$ peut être facilement encodée dans une instance de $\mathcal{D}_{single}$ en dupliquant chaque maintenance autant de fois qu'il y a de périodes. La complexité de cette transformation est un polynôme du nombre de périodes. On en déduit alors le résultat suivant.

Corollaire 2.3.4. *$\mathcal{D}_{single}$ est NP-complet au sens fort.*

NP-complétude de $\mathcal{D}_{single}$ pour $G = 1$ Dans le cas $\mathcal{D}_{single}$, on peut également prouver avec une réduction différente que le problème reste NP-complet au sens fort lorsqu'il n'y a qu'un seul composant.

Proposition 2.3.5. *$\mathcal{D}_{single}$ est NP-complet au sens fort, y compris si $G = 1$*

Démonstration. Ce résultat est également prouvé par réduction de 3-PARTITION. Soit I une instance de 3-PARTITION, notons f l'application transformant une instance de 3-PARTITION en une instance de $\mathcal{D}_{single}$ de la manière suivante. L'horizon de temps est fixé à $T = 2l$. On considère $M = 3l$ opérations de maintenances telles qu'il existe une bijection $\mu : a \to \mathcal{M}$ entre l'ensemble des items et l'ensemble des opérations de maintenance. Pour chaque item $a \in A$, l'opération de maintenance $\mu(a)$ régénère l'unique composant d'un montant égal à $r_{1\mu(a)} = 100s(a)/K$ et cause une perte de capacité de production $p_{\mu(a)} = 1/3$. Le coût fixe de maintenance vaut $s = 1$, et tous les autres coûts sont nuls. La capacité de production q_t est constante et égale à 1 sur tout l'horizon de temps. Une unité de production dégrade le niveau de santé de l'unique composant par $w_1 = 100$. Enfin, la santé initiale est égale à $H_{1,0} = 0$.

Le problème de décision que l'on veut résoudre est le suivant : "Existe-t-il un planning de maintenance tel que le coût soit inférieur ou égale à l ?". Le principe de cette construction repose sur le fait qu'il n'existe pas d'état inactif dans la modélisation proposée : la machine est soit en activité, soit en maintenance. La machine est alors forcée en maintenance sur les périodes impaires, et en production sur les périodes paires. On note $Y_{\mathcal{D}_{single}}$, l'ensemble des instances "oui" de $\mathcal{D}_{single}$. Montrons que $I \in Y_{3P} \iff f(I) \in Y_{\mathcal{D}_{single}}$.

$\Rightarrow$ Supposons $I \in Y_{3P}$, notons comme précédemment A_{ij} le j-ième item du i-ième sous-ensemble de la partition. On considère le plan de maintenance suivant : $P = (P_t)_{t\in\{1\dots 2l\}}$ où $\forall t \in \{1 \dots l\},\ P_{2t} = \{A_{t1}, A_{t2}, A_{t3}\}, P_{2t+1} = \emptyset$. La positivité de la santé et la contrainte sur la perte de capacité de production totale par période sont vérifiées à chacune des périodes du planning. Donc P est faisable avec valeur d'objectif l. D'où $f(I) \in Y_{\mathcal{D}_{single}}$. Ce plan de maintenance est illustré sur la Figure 2.4. Ici, l'axe des ordonnées et les pointillés bleus représentent la santé de l'unique composant, tandis que les rectangles gris représentent la régénération de chaque maintenance.

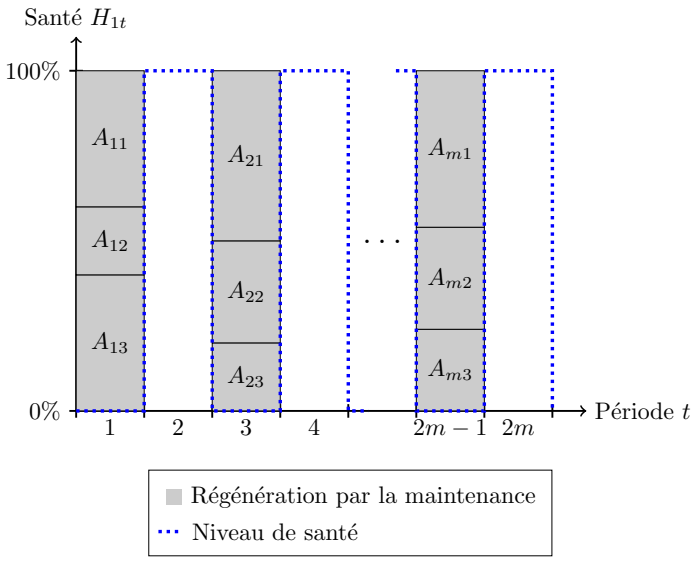


FIGURE 2.4 – Réduction de 3-PARTITION dans $\mathcal{D}_{single}$

$\Leftarrow$ Supposons à présent que I soit une instance de 3-PARTITION telle que $f(I) \in Y_{\mathcal{D}_{single}}$. Il existe un plan de maintenance tel que la valeur d'objectif soit l. Par hypothèse, l'objectif peut être exprimé de la manière suivante :

$$COUTS = \sum_{t\in\mathcal{T}} s\mathbb{1}_{P_t\neq\emptyset} = s|\{t|P_t \neq \emptyset\}|$$

Puisque $s = 1$, il peut donc y avoir au plus l périodes contenant au moins une maintenance. Par conséquent, il y a moins l périodes sans maintenances, où la santé diminue de $w_1 q_t = 100$. Une borne inférieure de la santé totale consommée tout au long de l'horizon de temps est $100l$. De plus, la régénération totale disponible en

raison du nombre limité de maintenances est $H_{1,0} + \sum_{a \in A} r_{1\mu(a)} = 100l$. Ainsi, la santé totale consommée est exactement $100l$, et chacune des activités de maintenance doit être planifiée. On a donc $3l$ maintenances à planifier sur l périodes, et au plus trois maintenances peuvent être planifiées par période puisque $\forall a \in A,\ p_{\mu(a)} = 1/3$. Il s'agit donc de faire des paquets de 3 maintenances, dont les sommes des régénérations soient exactement égales à 100, ce qui correspond à une solution de 3-PARTITION. D'où $I \in Y_{3P}$.

□

Ainsi $\mathcal{P}_{single}$ est NP-difficile au sens fort, y compris pour un seul composant.

2.3.2 Programmation Linéaire en Nombres Entiers

Nous avons établi dans la section précédente que les deux variantes considérées de notre problème sont NP-complet au sens fort. Cela signifie que dans leur version générale, il n'existe ni algorithmes polynomiaux ni pseudo-polynomiaux permettant de les résoudre, à moins que $P = NP$. En raison de ces résultats, et de la proximité avec les problèmes de lot-sizing, des approches basées sur la programmation mathématique sont particulièrement adaptées pour la résolution du problème général.

Dans cette section, nous proposons une modélisation de programmation linéaire en nombres entiers. Le modèle fait intervenir quatre familles de variables de décision, résumées dans la Table 2.3.

La variable $X_{gt} \in [0, 100]$ décrit la quantité de régénération reçue par le composant g au cours de la période t, et est équivalente à la variable décrivant la quantité de production dans les modèles de lot-sizing. La variable $Y_{mt} \in \{0, 1\}$ correspond à la décision d'effectuer ou non la maintenance m à la période t. De même, la variable $Z_t \in \{0, 1\}$ correspond à la décision d'effectuer ou non au moins une maintenance au cours de la période t, quelque qu'elle soit. Ces deux familles de variables ont un fonctionnement similaire aux variables d'activation des modèles de lot-sizing, elles sont corrélées aux variables de régénération par les contraintes d'activation et interviennent dans l'expression de l'objectif économique. Le niveau de santé d'un composant g à un instant t est représenté par la variable de décision $H_{gt} \in [0, 100]$, qui est analogue à la variable de stockage des modèles de lot-sizing. Enfin, la famille de variables $B_{gmt} \in [0, 100]$ correspond au produit des variables H_{gt} et Y_{mt}, et est utilisée pour la linéarisation du terme CRE dans l'expression de l'objectif de consommation de ressources.

TABLE 2.3 – Variables de décision

$X_{gt} \in [0, 100]$	Régénération sur le composant g au cours de la période t
$Y_{mt} \in \{0, 1\}$	1 si la maintenance m est réalisée pendant la période t, 0 sinon
$Z_t \in \{0, 1\}$	1 si au moins une activité de maintenance quelconque est réalisée lors de la période t, 0 sinon
$H_{gt} \in [0, 100]$	Santé du composant g au début de la période t
$B_{gmt} \in [0, 100]$	Produit de H_{gt} et de Y_{mt}

$$\min. \sum_{t=1}^{T} sZ_t + \sum_{m=1}^{M}\sum_{t=1}^{T}(c_m + \rho q_t p_m)Y_{mt} \tag{2.12}$$

$$\min. \sum_{t=1}^{T}\sum_{m=1}^{M} \alpha_m Y_{mt} + \sum_{t=1}^{T}\sum_{g=1}^{G} \beta_g q_t \left(100 - H_{gt} + \sum_{m=1}^{M} p_m(B_{gmt} - 100Y_{mt})\right) \tag{2.13}$$

$$X_{gt} + H_{gt} \geq H_{gt+1} + q_t w_g (1 - \sum_{m=1}^{M} p_m Y_{mt}) \qquad \forall t \in \mathcal{T}\backslash T,\ \forall g \in \mathcal{G} \tag{2.14}$$

$$X_{gt} \leq \sum_{m=1}^{M} r_{gm} Y_{mt} \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G} \tag{2.15}$$

$$1 - \sum_{m=1}^{M} p_m Y_{mt} \geq 0 \qquad \forall t \in \mathcal{T} \tag{2.16}$$

$$Y_{mt} \leq Z_t \qquad \forall t \in \mathcal{T},\ \forall m \in \mathcal{M} \tag{2.17}$$

$$H_{g1} = H_{g,0} \qquad \forall g \in \mathcal{G} \tag{2.18}$$

$$H_{gt} + X_{gt} \leq 100 \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G} \tag{2.19}$$

$$0 \leq B_{gmt} \leq 100 Y_{mt} \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G},\ \forall m \in \mathcal{M} \tag{2.20}$$

$$0 \leq H_{gt} - B_{gmt} \leq 100(1 - Y_{mt}) \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G},\ \forall m \in \mathcal{M} \tag{2.21}$$

$$B_{gmt} \geq 0 \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G},\ \forall m \in \mathcal{M} \tag{2.22}$$

$$X_{gt}, H_{gt} \geq 0, \qquad \forall t \in \mathcal{T},\ \forall g \in \mathcal{G} \tag{2.23}$$

$$Y_{mt}, Z_t \in \{0, 1\} \qquad \forall t \in \mathcal{T},\ \forall m \in \mathcal{M} \tag{2.24}$$

L'équation (2.12) est l'objectif de minimisation des coûts économiques. Il s'agit de la linéarisation des équations (2.5), (2.6), (2.7) et (2.8). De même, l'équation (2.13) est la linéarisation de l'objectif de minimisation de consommation de ressources, introduit par les équations (2.9), (2.10) et (2.11). En particulier, la transposition directe de l'équation (2.11) introduit un produit de variables. Pour préserver la linéarité, dans l'expression de l'objectif (2.13), le produit est remplacé par une variable représentant la valeur du produit, tel qu'imposé par les contraintes additionnelles (2.20), (2.21) et (2.22).

L'équation (2.14) est la formulation linéaire de la contrainte de conservation de la santé, introduite par l'équation (2.3). Le minimum de l'expression initiale est linéarisé en remplaçant l'égalité par une inégalité et en introduisant une nouvelle contrainte (2.19) bornant par 100 la somme de la santé et de la régénération. En comparaison de la contrainte de conservation classique des problèmes de lot-sizing, le terme de droite, analogue au terme de demande, dépend de la capacité effective de production, qui dépend elle-même de la décision de maintenance, et donc des variables Y_{mt}. Cette dépendance n'est pas classique en lot-sizing.

Les contraintes d'activation, liant la variable de régénération X_{gt} respectivement aux variables de Y_{mt} et Z_t sont exprimées dans les équations (2.15) et (2.17). En particulier, l'équation (2.15) impose que la valeur de régénération appliquée au composant soit inférieure ou égale à la régénération théorique des maintenances planifiées au cours de la période. Cette inégalité rend possible la perte de régénération, quand la somme des régénérations apportées par les maintenances entraîne un dépassement du plafond de santé 100 sans violation de la contrainte (2.19). L'équation (2.16) impose qu'il n'est pas possible de perdre plus de capacité de production que disponible et limite ainsi le nombre de maintenances autorisées sur une période.

L'équation (2.18) initialise la santé de chaque composant.

Enfin les équations (2.23) et (2.24) définissent les domaines de chacune des variables de décision. Au total, ce PLNE fait intervenir $O(GMT)$ variables réelles, $O(MT)$ variables binaires et $O(GMT)$ contraintes.

2.4 Conclusion

Dans ce chapitre, nous avons introduit un problème de planification de la maintenance au niveau tactique, où l'état de santé des composants de l'équipement sont suivi à l'aide d'un indicateur de santé. Nous montrons que ce problème est NP-difficile au sens fort, et nous avons vu que sous certaines hypothèses, ce problème partage de nombreuses structures et contraintes avec les problèmes de lot-sizing rencontrés en planification de la production. En se basant sur cette observation, nous avons proposé une formulation de PLNE bi-critère pour ce problème. La résolution de ce PLNE multi-critère fait l'objet du chapitre suivant du manuscrit.

Chapitre 3

Résolution et approche multi-critère

Sommaire

Dans le chapitre précédent, nous avons introduit un problème de planification de la maintenance au niveau tactique, où la dégradation de l'équipement est décrite par un indicateur de santé. Ce problème fait intervenir deux objectifs, un objectif classique de minimisation des coûts économiques et un objectif de minimisation de la consommation de ressources. Une modélisation de programmation linéaire en nombres entiers bi-critère a été proposée. Dans ce chapitre, nous nous intéressons à sa résolution et à la prise en compte de la dimension multi-critère du problème.

3.1 Contexte

Nous nous concentrons plus spécifiquement dans ce chapitre sur la prise en compte de l'objectif de consommations de ressources introduit dans le chapitre 2, et dont nous rappelons la définition ici. La définition précise du problème étudié dans ce chapitre et les notations utilisées ont été introduites dans la section 2.1 du chapitre précédent.

$$RESS = CRM + CRE \tag{2.9}$$

Le terme CRM correspond à la consommation de ressources liée à la maintenance. Chaque maintenance m planifiée cause une consommation de ressource fixe α_m, soit :

$$CRM = \sum_{t \in \mathcal{T}} \sum_{m \in P_t} \alpha_m \tag{2.10}$$

Le terme CRE correspond à la consommation de ressources liée à l'exploitation de l'équipement usé et fait intervenir la santé de l'équipement dans son expression :

$$CRE = \sum_{t=1}^{T} \sum_{g=1}^{G} \beta_g q_t (100 - H_{gt})(1 - \sum_{m=1}^{M} p_m \mathbb{1}_{m \in P_t}) \tag{2.11}$$

où $\beta_g \in \mathbb{R}_+$ est un coefficient de proportionnalité. Nous revenons dans cette section sur le positionnement bibliographique de cette contrainte et la démarche adoptée pour la prise en compte de cet objectif. Nous présentons ensuite, dans la section 3.2 une première approche basée sur une contrainte de budget, puis dans la section 3.3 une approche basée sur le calcul du front de Pareto.

3.1.1 Considérations environnementales et lot-sizing multi-critère

En raison du gain d'intérêt récent pour la question climatique, de nombreuses études portent sur la prise en compte de ces thématiques en lot-sizing. Suzanne et al. (2020) présentent une revue très complète sur l'état de l'art de l'économie circulaire en planification de la production. Nous nous concentrons ici plus spécifiquement sur la consommation de ressources et les émissions. La littérature se concentre essentiellement sur la consommation d'énergie et les émissions de gaz à effet de serre. Plusieurs approches existent pour modéliser ces effets.

Du côté des émissions, les approches consistent essentiellement à imposer des plafonds d'émissions sous forme de contraintes. Absi et al. (2013) investiguent quatre modélisations pour ce type de contrainte : contraintes périodiques, contraintes sur les émissions cumulées,

contrainte globale sur l'horizon de temps, contrainte sur les émissions cumulées sur un horizon glissant. Retel Helmrich et al. (2015) proposent une contrainte globale plus générale et générique pour la modélisation des émissions. P. He et al. (2015) étudient quant à eux le problème de lot-sizing dans le contexte des marchés carbone. À notre connaissance, les approches multi-objectifs pour les émissions ont été très peu explorées à l'heure actuelle : on peut citer les travaux de Azadnia et al. (2015) et de K. Shaw (2017) qui s'intéressent conjointement au problème du choix du fournisseur et d'attribution des commandes et au problème de lot-sizing en présence d'un second objectif de minimisation des émissions.

En ce qui concerne la consommation d'énergie, certaines approches consistent à introduire les coûts énergétiques dans l'objectif. Giglio et al. (2017) introduisent une composante supplémentaire dans l'objectif économique classique du lot-sizing pour modéliser un coût énergétique. O. Masmoudi et al. (2017) intègrent également la consommation d'énergie dans les coûts économiques et imposent une contrainte sur la puissance disponible à chaque période. Wichmann et al. (2019) étudient le cas où les coûts énergétiques dépendent de la période. Hajej et Rezg (2020) utilisent des fonctions de coût énergétiques complexes dépendant de la quantité produite et du nombre de machines mises en route pour un problème de lot-sizing avec maintenance. On retrouve également une approche similaire à celles utilisées pour les émissions, consistant à imposer des plafonds de consommation au moyen de contraintes. Rapine et al. (2018) modélisent une consommation d'énergie dépendante de la quantité produite et des machines démarrées à chaque période. La consommation d'énergie à chaque période doit rester inférieure la quantité d'énergie disponible. À notre connaissance, il n'existe pas d'articles proposant une approche multi-objectif pour l'optimisation conjointe des coûts économiques et de la consommation d'énergie en lot-sizing.

3.1.2 Démarche de résolution

Pour notre problème, nous considérons deux approches de l'aspect multi-critère. La première approche consiste à traiter le critère de consommation de ressources comme une contrainte globale, similairement à certaines approches de la littérature évoquée précédemment. L'étude de la sensibilité de l'objectif économique aux variations de la consommation de ressource maximum permet d'obtenir des premiers résultats sur la nature du compromis entre les deux objectifs. Cette approche est traitée en détail dans la section 3.2, et a fait l'objet d'une publication en conférence internationale (Foussard et al., 2021).

Dans un second temps, nous adoptons une approche d'optimisation multi-critère, dans la perspective de développer un outil d'aide à la décision. Les fronts de Pareto sont approximés en combinant la méthode ε-contrainte avec un solveur de programmation linéaire en nombre entiers. Une analyse des performances de la méthode et de la répartition des points difficiles à calculer sur le front de Pareto nous permet ensuite de dériver une procédure d'aide à la décision dédiée permettant de fournir un ensemble diversifié de solutions au preneur de décision. Cette approche est traitée dans la section 3.3 et fait l'objet d'un article soumis en revue.

3.2 Approche mono-objectif avec contrainte de budget

Dans une première approche, nous utilisons une technique de modélisation typique de la littérature du lot-sizing pour la prise en compte de deux objectifs. L'objectif économique reste un objectif à minimiser, tandis que l'objectif de consommation de ressources est transformé en contrainte de budget. La quantité de ressources disponibles est donnée en entrée du problème, et on cherche à optimiser le coût économique en respectant cette contrainte sur la consommation de ressources.

3.2.1 Contrainte de budget

Pour cette section, seul le terme de consommation de ressources liée à l'exploitation de l'équipement usé est pris en compte. Autrement dit, on fait l'hypothèse $\forall m \in \mathcal{M},\ \alpha_m = 0$. La contrainte de budget s'écrit alors de la manière suivante :

$$RESS = CRE = \sum_{t=1}^{T} \sum_{g=1}^{G} \beta_g q_t (100 - H_{gt})(1 - \sum_{m=1}^{M} p_m \mathbb{1}_{m \in P_t}) \leq BUDGET \tag{3.1}$$

En pratique, le fait que les opérations de maintenance n'induisent pas de coûts de maintenance incite à effectuer des maintenances dans le seul but d'empêcher la production. Le terme CRM a été introduit au terme de cette première étude afin d'éviter ce cas extrême irréaliste.

3.2.2 Environnement expérimental

Le modèle a été implémenté en OPL à l'aide de IBM ILOG CPLEX Optimization Studio 12.10. Toutes les expérimentations ont été réalisées sur un ordinateur portable doté de 16 Go de RAM et d'un processeur Intel Core i7-10610U CPU @ 1.80GHz × 8, tournant sur Ubuntu 20.04.2 LTS. Le temps maximum de résolution a été fixé à 10 minutes par instance dans le solveur.

Les expérimentations ont été conduites sur des jeux de données générés aléatoirement en se basant sur les caractéristiques des lave-linges. L'objectif est de pouvoir évaluer le comportement du modèle et les performances de résolution pour des instances de taille réaliste. Les instances considérées sont de taille variable, avec un horizon de temps $T = 52$ représentant un an découpé en périodes d'une semaine, ce qui correspond à un horizon de temps typique du niveau tactique de décision.

On considère trois familles d'instances correspondant à des profils de maintenance différents :

- $F1$: Chaque composant a une maintenance dédiée, régénérant entièrement sa santé.
- $F2$: Chaque composant a une maintenance dédiée, apportant une régénération de 50 au composant.
- $F3$: $M = 2G$ maintenances non-ciblées touchant plusieurs composants simultanément, réparties en deux types : G maintenances régénérant 20% des composants de 80, et G maintenances régénérant 80% des composants de 20. Chaque composant est concerné par au moins une maintenance de chaque type.

Les instances sont générées initialement pour $G = 8$ composants, puis des sous-instances pour $G = 4$ et $G = 6$ sont extraites à partir d'elles. Ces nombres de composant permettent généralement de couvrir les composants critiques d'un lave-linge de lavomatique. Les coûts et perte de capacités liées à la maintenance sont tirés avec des lois uniformes de paramètres proportionnels à la régénération totale apportée par la-dite maintenance. Si $S_m = \sum_{g \in \mathcal{G}} r_{gm}$, alors :

— $c_g \sim U([0.5 \cdot S_m, 1.5 \cdot S_m])$
— $p_m \sim U([0.004 \cdot S_m, 0.006 \cdot S_m])$

La valeur du paramètre pour p_m a été choisie de telle sorte qu'une opération de maintenance cause une perte équivalente à 30% à 60% de la capacité de production sur une période. Le coût fixe s est fixé à 10% du coût variable de maintenance moyen. La capacité de production q_t prend deux valeurs : 0.75 représentant une période creuse, et 1 représentant une période pleine. L'état de santé initial des composants de l'équipement est obtenu par tirage de loi uniforme sur l'intervalle $[50, 100]$. Afin d'avoir des composants ayant des vitesses de dégradation variables, l'usure par unité de production de chaque composant est obtenu par tirage uniforme sur l'intervalle $[2, 10]$. Enfin, le rendement d'une unité de production ρ est fixé à 1. Ainsi les coûts de maintenances et coûts d'opportunités se situent dans le même ordre de grandeur. Enfin, les coefficients β_g sont tirés de manière uniforme dans l'intervalle choisi arbitrairement $[0.5, 1.5]$.

Dans un premier temps, nous évaluons l'impact de la taille des instances sur les performances de résolution lorsque la contrainte de budget est inactive. Puis, l'impact de la contrainte sur la résolution est évalué dans un second temps.

3.2.3 Impact de la taille des instances

Afin d'évaluer l'impact de la nature et de la taille des instances sur les performances de résolution, les résultats obtenus sur les instances générées ont été regroupés par nombre de composants et profils de maintenance dans la Table 3.1. Pour ces expérimentations, la valeur du budget est fixée à une valeur très grande pour que la contrainte devienne inactive. Les caractéristiques des instances sont décrites dans les deux premières colonnes. Le gap moyen, calculé comme le ratio de la somme des gaps absolus par la somme des bornes supérieures, est présenté dans la troisième colonne. Les gaps minimum et maximum sont présentés dans la quatrième et la cinquième colonne. Enfin, le nombre d'instances pour lesquels l'optimalité a été prouvée est indiqué dans la sixième colonne.

TABLE 3.1 – Impact de la taille des instances et du type de maintenance

	G	Gap Moyen	Gap min	Gap max	# opt
F1	4	0.4%	-	2.4%	9/10
	6	7.9%	-	14.5%	2/10
	8	15.1%	7.7%	20.1%	0/10
F2	4	0.9%	-	7.1%	8/10
	6	5.6%	2.3%	11.0%	0/10
	8	8.0%	5.5%	12.2%	0/10
F3	4	0.1%	-	1.1%	9/10
	6	0.3%	-	2.7%	9/10
	8	1.3%	-	7.7%	7/10

On observe que la nature des maintenances a un impact important sur l'efficacité de la résolution du modèle : en effet, il apparaît que les instances ayant les profils de maintenance $F1$ et $F2$ sont rarement résolues à l'optimum pour $G = 6$ ou $G = 8$, et que les gaps sont importants. Au contraire, pour les instances ayant pour profil de maintenance $F3$, les solutions obtenues au terme de la résolution sont souvent optimales ou ont une bonne garantie, y compris quand $G = 8$. De manière générale, on observe que l'augmentation du nombre de composants se traduit par une baisse de la qualité des solutions obtenues au terme des 10 minutes. Cela est cohérent avec l'augmentation du nombre de variables et de contraintes dans le modèle liée à l'augmentation du nombre de composants.

3.2.4 Impact de la contrainte de budget

Nous nous intéressons à présent à l'impact de la contrainte de budget lorsqu'elle est active. Pour ce faire, nous reprenons les instances avec $G = 4$ et le profil de maintenance $F1$. Nous récupérons les valeurs obtenues pour la consommation de ressources dans la section précédente, et les utilisons comme base de comparaison. Ces instances sont résolues une nouvelle fois, en imposant cette fois un budget égal à 50% et 75% de cette valeur de base. Les résultats obtenus sont présentés dans la Table 3.2. La valeur du budget est précisé dans la première colonne, et nous retrouvons dans la deuxième et cinquième colonne les indicateurs de gap moyen et nombre d'instances résolues à l'optimum utilisées précédemment. Les augmentations moyennes en pourcentage de la valeur de l'objectif et du nombre de maintenances planifiées par rapport à l'instance de base sont présentées dans la troisième et la quatrième colonne respectivement.

TABLE 3.2 – Impact de la contrainte de budget

Budget	Gap moyen	Augm. moy. obj.	Augm. moy. maint.	# opt
50%	11.7%	+48.5%	+57.6%	0/10
75%	1.9%	+7.1%	+12.0%	8/10
100%	0.6%	+0.0%	+0.0%	9/10

En regardant l'évolution du gap moyen et du nombre d'instances résolues à l'optimum, il

apparaît que plus le budget de consommation de ressources est contraignant, plus la résolution des instances est difficile. La qualité des solutions obtenues décroît.

Par ailleurs, on observe que la contrainte de budget et l'objectif économique agissent de manière antagoniste. Plus le budget décroît, plus les coûts économiques sont élevés. Dans le cas 75%, le phénomène est peu visible. En effet, dans la résolution initiale, la consommation de ressources ne subit aucune optimisation. Il est alors possible d'améliorer cette solution initiale en cherchant des symétriques ayant une consommation de ressources moins importante sans compromettre la valeur de l'objectif économique. Ainsi, une faible réduction ne nécessite pas une forte augmentation des coûts économiques. En revanche, l'augmentation est bien visible dans le cas 50%.

Les résultats obtenus sont conformes aux attentes : la diminution des consommations de ressources nécessite de maintenir l'équipement dans un meilleur état, et donc de planifier davantage de maintenances, ce qui représente des coûts économiques plus importants. L'étude de ce compromis est approfondie dans la suite, avec une approche bi-critère du problème.

3.3 Approche bi-critère

La première approche proposée dans la section 3.2 consistait à poser l'objectif économique comme objectif principal à minimiser, en imposant le second objectif comme contrainte de budget. Cette méthode a permis de donner une première évaluation du compromis entre les deux objectifs, mais reste toutefois assez limitée en comparaison des approches multi-critères de la littérature.

Dans cette section, nous étudions plus en profondeur le compromis entre les deux critères d'optimisation considéré. Nous analysons ensuite les fronts de Pareto obtenus sur un jeu d'instances générées aléatoirement, et dérivons une procédure d'aide à la décision permettant de fournir un ensemble restreint de solutions diversifiées et représentatives au preneur de décision. Les principales notions d'optimisation multicritère utiles à la compréhension de ce qui suit ont été introduites dans le chapitre 1.

Les travaux présentés dans cette section sont soumis dans un article de revue.

3.3.1 Approximation du front de Pareto par la méthode ε-contrainte

Il existe de nombreuses techniques dans la littérature permettant de calculer ou d'approcher le front de Pareto. Parmi les méthodes les plus utilisées en pratique, on peut relever en particulier la combinaison convexe de critères, qui consiste à étudier la somme pondérée de tous les objectifs et à faire varier les poids, la méthode de l'analyse paramétrique qui consiste à chercher les optima de Pareto sur des courbes de niveaux dans l'espace des critères, ou encore l'approche goal programming qui consiste à fixer un point de référence comme un objectif à atteindre dans l'espace des critères, puis à chercher les points qui minimisent une distance avec le point de référence.

On s'intéresse ici plus particulièrement à la méthode ε-contrainte, qui repose sur une idée similaire à la contrainte de budget : ne garder qu'un seul objectif et transformer les autres en contraintes. La méthode consiste à faire varier la borne sur les objectifs contraints afin d'obtenir les optima de Pareto de manière itérative. Le choix de cette méthode est motivé

par plusieurs considérations : d'une part, il s'agit d'une des méthodes les plus utilisées et documentées pour le calcul pratique de fronts de Pareto. De plus, cette méthode ne nécessite pas de normalisation ou de comparaisons directes entre les objectifs. Nos deux objectifs n'ont pas la même dimension physique, ce qui rend cette méthode particulièrement avantageuse.

La méthode est détaillée pour le cas bi-objectif, tel que considéré pour notre problème.

On définit le problème ε-contraint de la manière suivante :

$$\begin{aligned} \min. \quad & Z_1(x) \\ \text{s.t.} \quad & Z_2(x) < \varepsilon \\ & x \in \mathcal{S} \end{aligned}$$

La méthode est initialisée en résolvant séparément les deux problèmes mono-objectifs, ce qui permet d'obtenir les points extrêmes x_1^* et x_2^*, puis on résout le problème ε-contraint en fixant $\varepsilon = Z_2(x_1^*)$ ce qui permet d'obtenir un nouveau point x_3^*, on réitère le procédé en posant à chaque itération $\varepsilon = Z_2(x_i^*)$ jusqu'à retomber sur x_1^*, ce qui signifie que tout le front de Pareto a été balayé. L'algorithme est détaillé sur la Figure 3.1.

Le problème ε-contraint, comme défini ci-dessus, n'est pas linéaire en raison de l'inégalité stricte. Pour cette raison, on introduit un pas d'échantillonnage Δ, qui est soustrait à ε à chaque itération. On obtient ainsi une approximation du front de Pareto. Plus le pas d'échantillonnage est grand, moins il y aura de points dans l'approximation. Cela n'empêche pas toutefois que la distance entre deux points sur le front soit supérieure à Δ, dans le cas où la contrainte n'est pas serrée.

3.3.2 Environnement expérimental

Les modèles ont été implémentés en OPL avec IBM ILOG CPLEX Optimization Studio 20.1.0.0. Toutes les expérimentations de cette section ont été réalisés sur un serveur Dell T640, doté de deux processeurs Intel Xeon Silver 4210R (2,4GHz, 10C/20T, cache 13,75Mo, 9,6GT/s, 100W, Turbo, HT) et de 128 Go de RAM, tournant sur Debian GNU/Linux 11 (bullseye). En pratique, nous avons limité CPLEX à 4 threads et 4 Go de RAM.

Les jeux de données utilisés ont été générés en suivant le protocole détaillé ci-après. Les instances sont disponibles à l'adresse suivante : https://doi.org/10.5281/zenodo.8091094

Pour toutes les instances, l'horizon de temps est fixé à $T = 52$, ce qui représente par exemple un an avec une granularité de l'ordre de la semaine. Nous considérons le cas de 1, 2, 4 et 8 composants. Pour la maintenance, deux profils sont considérés. Dans le premier cas, chaque composant a une opération de maintenance parfaite ciblée dédiée, le régénérant à son niveau de santé maximum 100. Dans le second cas, chaque composant a une opération de maintenance imparfaite dédiée, régénérant sa santé de 80 et celle de tous les autres composants de 20. 5 instances ont été générées pour chaque combinaison nombre de composants / profils de maintenance, pour un total de 40 instances.

Les autres paramètres sont obtenus par des tirages indépendants suivant une loi uniforme sur les intervalles suivants : $s \in [0, 100]$, $c_m \in [0, 1000]$, $\rho \in [0, 1000]$, $q_t \in [0.5, 1]$, $p_m \in [0, 1]$, $w_g \in [1, 20]$, $H_{g,0} \in [0, 100]$. Les intervalles ont été choisis de manière à ce que le coût d'opportunité et le coût de maintenance impactent plus fortement l'objectif de coût économique

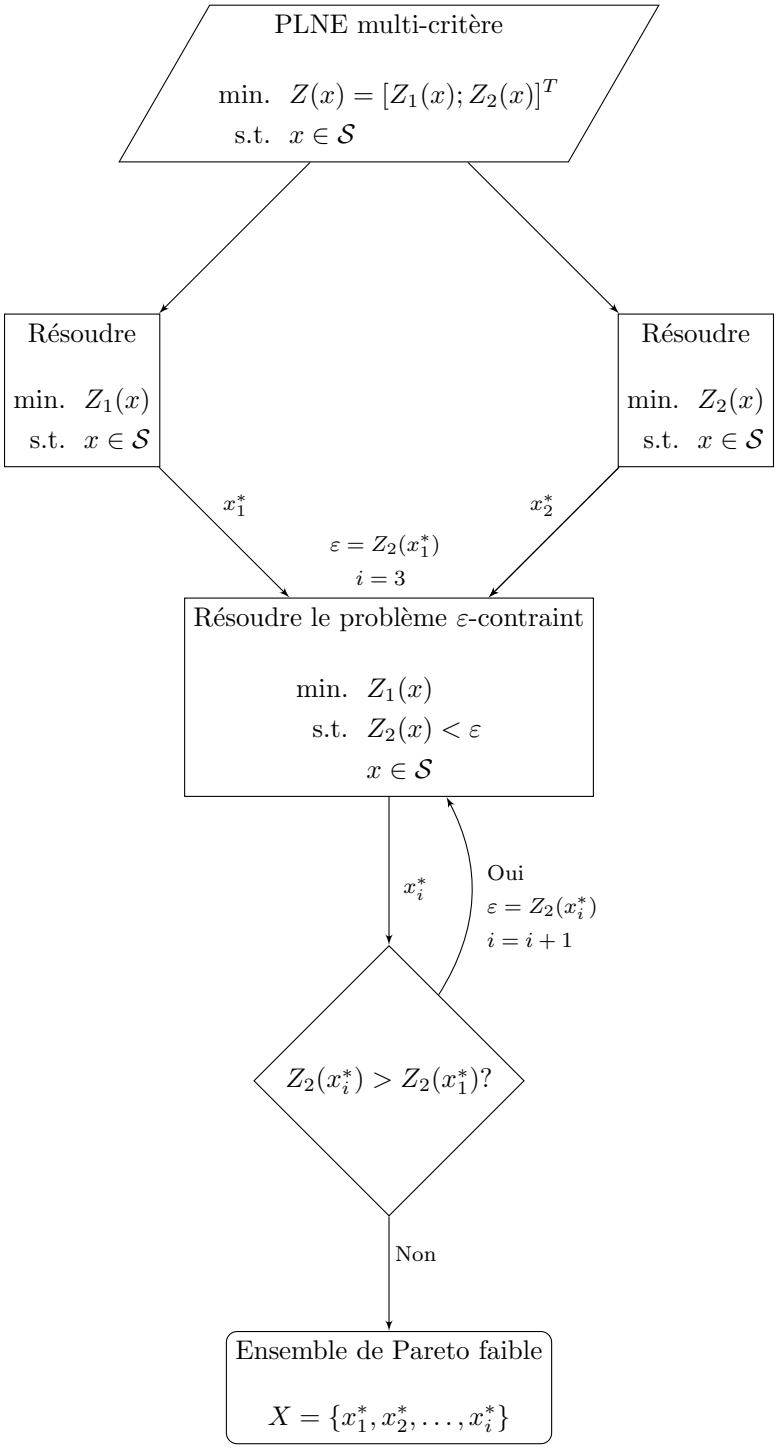


FIGURE 3.1 – Méthode ε-contrainte. *Schéma extrait et adapté de la thèse de Martinez (2020)*

que les coûts fixes, et qu'il soit nécessaire de planifier régulièrement des maintenances, voire plusieurs au cours de la même période. Les valeurs des coefficients pour l'objectif de consommation de ressources α_m et β_g sont fixés arbitrairement à 1.

Au cours de premiers tests, il est apparu que le modèle ε-contraint peut nécessiter un temps de calcul important pour obtenir des solutions avec une bonne garantie dès que le nombre de composants est plus grand que $G = 2$. Dans ces conditions, il n'est possible de calculer efficacement un front de Pareto, qui nécessite de résoudre un grand nombre de fois ce modèle. Cela nous a conduits à étudier des instances plus petites afin de pouvoir calculer des premiers fronts et identifier où se situe la difficulté, et comment la surmonter pour parvenir à fournir une méthodologie d'aide à la décision efficace, y compris pour les plus grandes instances.

Pour ce faire, une des instances générées pour $G = 1$ avec maintenance parfaite a été modifiée pour construire trois nouvelles instances avec des horizons de temps plus faibles : $T = 10$, $T = 20$ et $T = 50$.

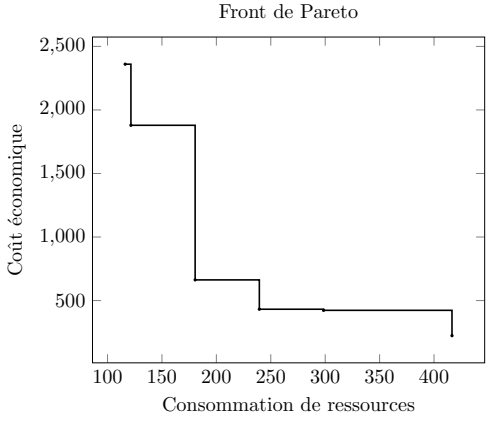
3.3.3 Analyse des fronts de Pareto

Dans un premier temps, nous étudions les performances de la méthode et l'allure des fronts de Pareto pour les trois instances réduites évoquées précédemment. Nous définissons la taille d'échantillonnage de la manière suivante : $\frac{Z_2(x_1)-Z_2(x_2)}{\Delta}$. Cette quantité est une borne supérieure sur le nombre de points que contiendra l'ensemble de Pareto au terme de l'exécution de la méthode ε-contrainte. Plus le pas d'échantillonnage est petit, plus le front de Pareto sera échantillonné. Chaque instance est résolue pour des tailles d'échantillon variables 10, 50 et 100. Pour ces premières instances, de petite taille, aucune limite de temps restrictive n'a été fixée pour le solveur et le modèle ε-contrainte est systématiquement résolu à l'optimum.

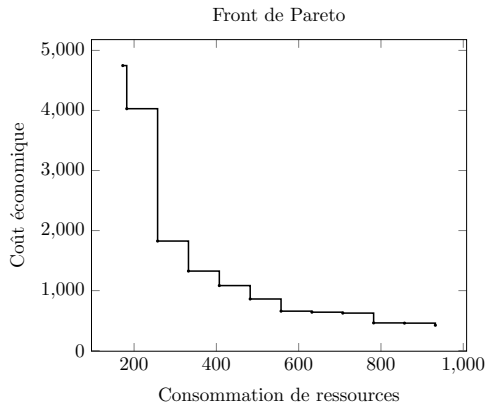
La Table 3.3, regroupe les valeurs obtenues pour l'hypervolume normalisé, le temps total de calcul et le pourcentage de solutions non-dominées en fonction de la taille de l'instance et de la taille d'échantillon. Figure 3.2 et Figure 3.3 représentent respectivement les fronts de Pareto obtenus pour les instances avec $T = 10$, $T = 20$ et $T = 50$. L'objectif économique est représenté sur l'axe vertical, et l'objectif économique sur l'axe horizontal.

T	Échantillonnage	Hypervolume	Temps de calcul	Solutions non-dominées
10	10	0.734	0.15s	50%
	50	0.780	0.71s	18%
	100	0.797	1.57s	13%
20	10	0.811	0.55s	100%
	50	0.836	2.40s	50%
	100	0.842	4.94s	37%
50	10	0.805	14.37s	92%
	50	0.847	62.00s	71%
	100	0.851	135.00s	60%

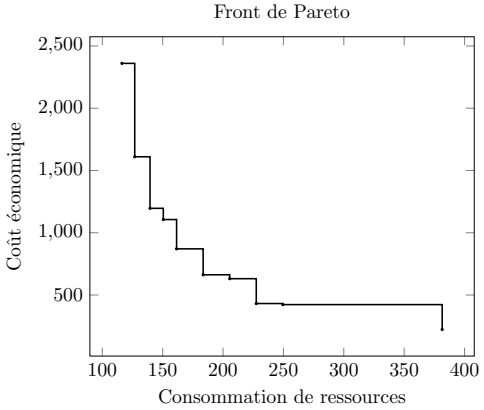
TABLE 3.3 – Résultats expérimentaux sur les petites instances test



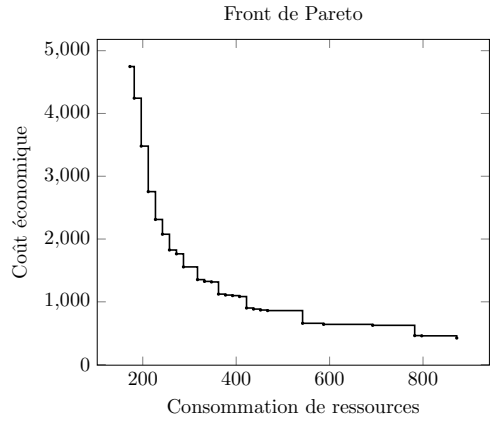
(a) $T = 10$, Échantillonnage : 10



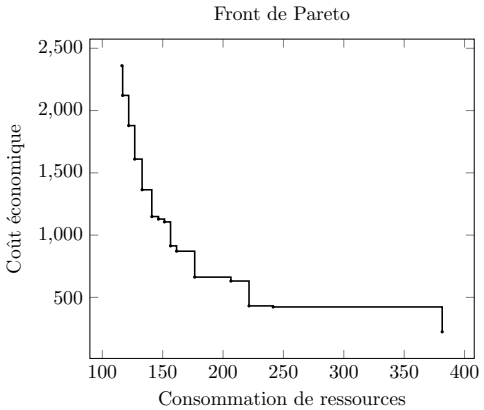
(b) $T = 20$, Échantillonnage : 10



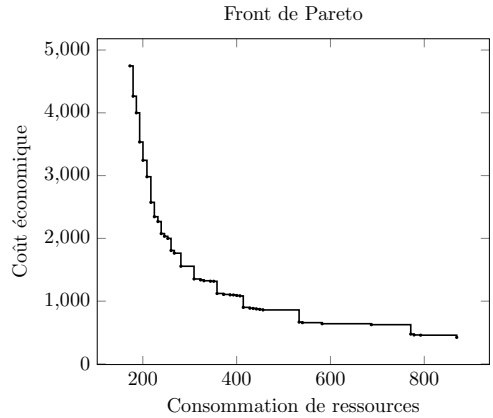
(c) $T = 10$, Échantillonnage : 50



(d) $T = 20$, Échantillonnage : 50

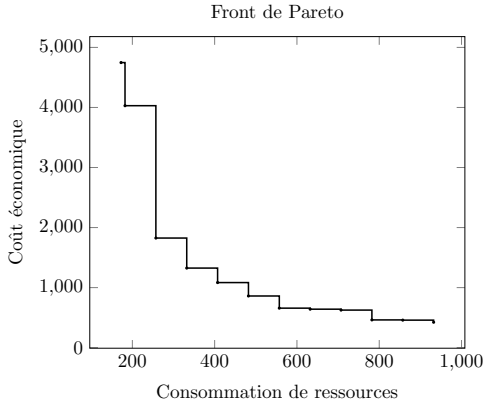


(e) $T = 10$, Échantillonnage : 100

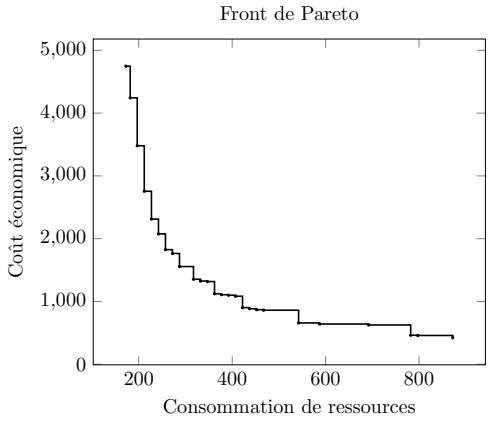


(f) $T = 20$, Échantillonnage : 100

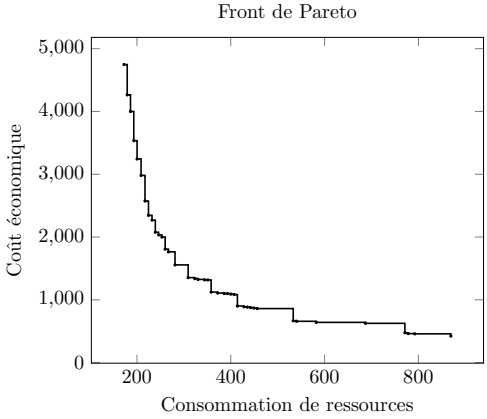
FIGURE 3.2 – Fronts de Pareto obtenus sur les petites instances test pour $T = 10$ et $T = 20$



(a) $T = 50$, Échantillonnage : 10



(b) $T = 50$, Échantillonnage : 50



(c) $T = 50$, Échantillonnage : 100

FIGURE 3.3 – Fronts de Pareto obtenus sur les petites instances test pour $T = 50$

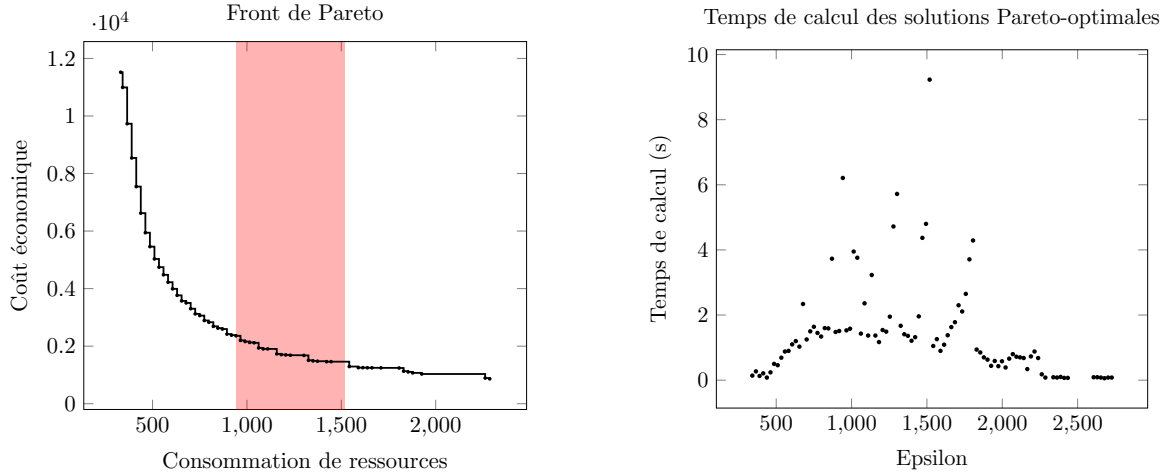
Concernant la taille d'échantillonnage, il est intéressant de comparer les hypervolumes et le nombre de solutions non-dominées entre les différents fronts obtenus afin de déterminer dans quelle mesure améliorer l'échantillonnage améliore leur qualité. Visuellement, on observe qu'y compris quand la taille d'échantillon vaut 10, on parvient à obtenir un ensemble de solutions diversifiées et représentatives du compromis. Une taille d'échantillon de 50 permet d'obtenir une représentation plus fidèle, on peut observer une amélioration de l'hypervolume pour les trois instances. On constate toutefois une détérioration significative du taux de solutions non-dominées, ce qui signifie qu'on calcule un nombre important de points peu intéressant. Par exemple, pour le cas $T = 20$, pour une taille d'échantillon de 10, toutes les solutions obtenues sont non-dominées et sont donc intéressantes du point de vue de l'aide à la décision. En comparaison, pour une taille d'échantillon de 50, seule la moitié d'entre elles, soit 25, le sont, ce qui reste toutefois intéressant. La taille d'échantillon de 100 ne permet d'améliorer que marginalement la qualité du front : l'hypervolume ne change pas de manière significative,

et le taux de solutions non-dominées se dégrade significativement à l'exception du cas $T = 50$.

On observe que la méthode tend à sur-échantillonner la partie du front correspondant aux solutions à faible coût économique tandis que la partie du front à faible consommation de ressources est sous-échantillonnée. Cela est particulièrement visible lorsqu'on a une faible taille d'échantillon. Ce problème est lié au choix d'un pas d'échantillonnage constant en termes de consommation de ressources, et pourrait être surmonté en envisageant une méthode d'échantillonnage différente.

Concernant le temps de calcul, on observe conformément aux attentes que l'évolution du temps de calcul est proportionnelle aux nombres de points. On constate également qu'augmenter la taille de l'instance augmente drastiquement le temps de calcul sur l'ensemble du front de Pareto. Cela confirme pour le cas d'instances de taille réaliste la nécessité de développer une méthode dédiée qui permette d'obtenir plus efficacement de bonnes solutions.

Nous nous intéressons également à la répartition des instances difficiles. La Figure 3.4 illustre la répartition des temps de calcul sur le front de Pareto pour l'instance $T = 20$ avec une taille d'échantillon égale à 100. Sur la figure de droite, le temps de calcul de chaque point est représenté en fonction de la valeur de ε correspondante. Il apparaît qu'un sous-ensemble marginal de points sur une région localisée du front sont nettement plus difficiles à calculer que les autres. La région représentée en rouge sur la figure de droite contient la totalité des points qui ont nécessité un temps de calcul supérieur à 5 secondes. On note toutefois qu'au sein de cette région, la plupart des points restent faciles à calculer. On retrouve ce type d'observation pour les autres instances considérées. Dans la suite, nous proposons une procédure qui s'appuie sur ces observations afin de fournir rapidement un ensemble de solutions diversifiées au preneur de décision.



(a) Région contenant toutes les instances difficiles

(b) Temps de calcul en fonction de ε

FIGURE 3.4 – Répartition des instances difficiles sur le front de Pareto pour $T = 20$ et taille d'échantillon 100.

3.3.4 Procédure dédiée

Nous avons vu précédemment que le calcul d'un front de Pareto pose un véritable défi sur le plan computationnel dès que les instances sont un peu grandes. On peut estimer qu'il n'est pas aberrant pour un équipement industriel complexe de suivre la santé de plusieurs composants critiques sur un horizon de temps de l'ordre de l'année. Ce type d'instance ne peut pas être résolu efficacement en appliquant l'approche proposée précédemment, en raison de la difficulté de certaines instances situées dans la région difficile, qui nécessitent des temps de calcul très importants pour obtenir une solution avec une bonne garantie d'optimalité. Pour palier à ce problème, nous proposons une approche différente dans une optique d'aide à la décision, visant à fournir un ensemble diversifié de solutions avec une bonne qualité d'optimisation. On ne cherche plus à résoudre à l'optimum chacune des instances correspondant à un point du front de Pareto, on se contentera de solutions ayant un gap d'optimalité faible, c'est-à-dire, proche de l'optimum. L'objectif est d'avoir une procédure qui, dans un temps limité, permet d'obtenir un ensemble de solutions représentatives du front de Pareto avec une bonne qualité d'optimisation.

On a constaté que seul un ensemble restreint d'instances dans une zone localisée du front de Pareto étaient difficiles à calculer, et que de plus, sur les expérimentations réalisées, on observe que les points à proximité de ces points difficiles ne sont pas nécessairement difficiles. L'approche adoptée consiste à sur-échantillonner le front de Pareto puis à écarter les points difficiles.

Plus formellement, les deux étapes de la procédure sont les suivantes :

1. Calculer un ensemble de Pareto faible par la méthode ε-contrainte pour une taille d'échantillon égale à 10, avec une limite de temps contraignante pour la résolution du modèle ε-contrainte.
2. Obtenir un sous-ensemble représentatif du front à l'aide d'une méthode de sélection.

Pour les expérimentations présentées ci-après, une limite de 15 minutes est imposée pour la résolution du modèle ε-contraint et la taille d'échantillon est fixée à 10. En comptant les deux points extrêmes calculés lors de l'initialisation de la méthode, le calcul du front de Pareto consiste à calculer au plus 12 points, soit un temps de calcul maximum de 4 heures.

Nous proposons deux méthodes pour sélectionner les solutions à fournir au preneur de décision. La première est une méthode visuelle : dans un premier temps, les points ayant un gap d'optimalité trop important sont écartés. Dans un second temps, on choisit environ 5 points sur le fronts de sorte à ce que les points soient distribués de manière à peu près uniforme sur le front. Il est toujours intéressant de fournir les points extrêmes du front de Pareto : ils sont en général faciles à calculer, car les modèles correspondants sont moins complexes puisqu'ils ne font intervenir qu'un seul des deux objectifs, et ils permettent également d'obtenir des bornes pour les valeurs atteignables pour les deux objectifs.

Cette méthode de sélection visuelle a été testée sur des instances du benchmark pour les cas de $G = 2$ et $G = 4$, dans le cas de la maintenance parfaite ou imparfaite. Les fronts obtenus peuvent être visualisés sur la Figure 3.5, la Figure 3.6 et la Figure 3.7. Sur ces figures, les différents points calculés sont numérotés de gauche à droite. Pour chacun de ces points, l'intervalle entre la valeur de la meilleure solution obtenue (borne supérieure) et la meilleure borne inférieure trouvée sont représentés en rouge. Plus l'intervalle est réduit, plus le gap d'optimalité est faible. Les points sélectionnés au terme de la procédure de sélection sont

représentés en bleu. Qualitativement, nous parvenons grâce à cette méthode sur les instances testées à obtenir des ensembles diversifiés de solutions. On retrouve toutefois une difficulté liée à la méthode d'échantillonnage : les points sont très diversifiés pour l'objectif de consommation de ressource, mais assez peu pour l'objectif économique. Le pas constant en abscisses fait que l'on a une forte redondance sur la partie droite du front, où la pente est faible et que la partie gauche du front, où la pente est forte est largement sous-échantillonnée. Une méthode d'échantillonnage différente permettrait de renforcer significativement la procédure d'aide à la décision.

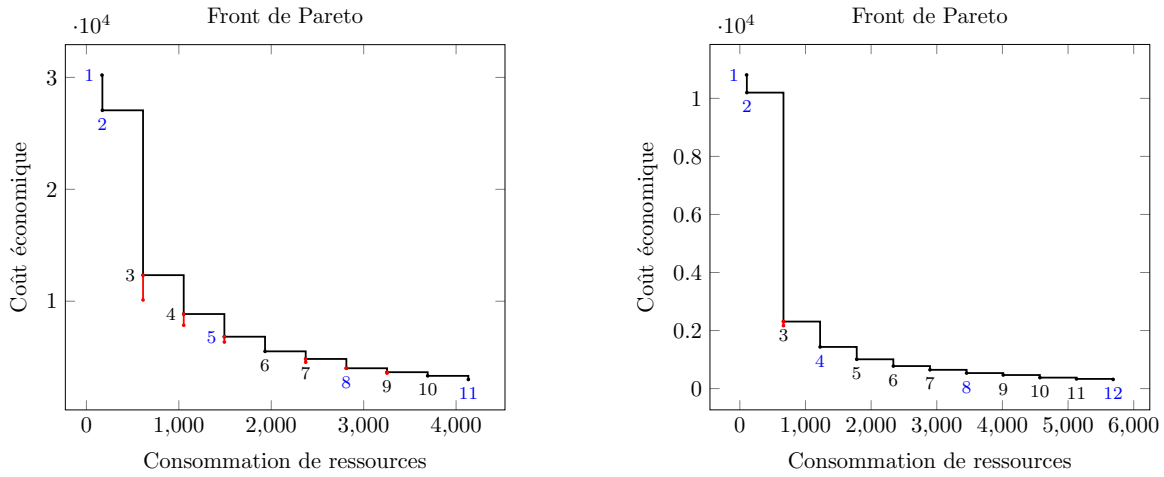


FIGURE 3.5 – Fronts de Pareto obtenus pour $G = 2$ et maintenance parfaite. Les points sélectionnés sont représentés en bleu.

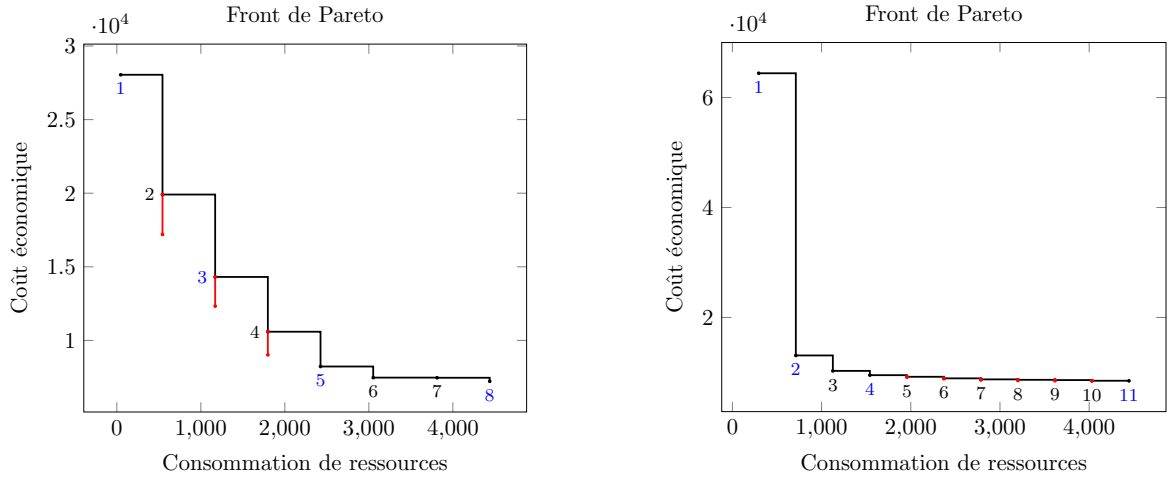


FIGURE 3.6 – Fronts de Pareto obtenus pour $G = 2$ et maintenance imparfaite. Les points sélectionnés sont représentés en bleu.

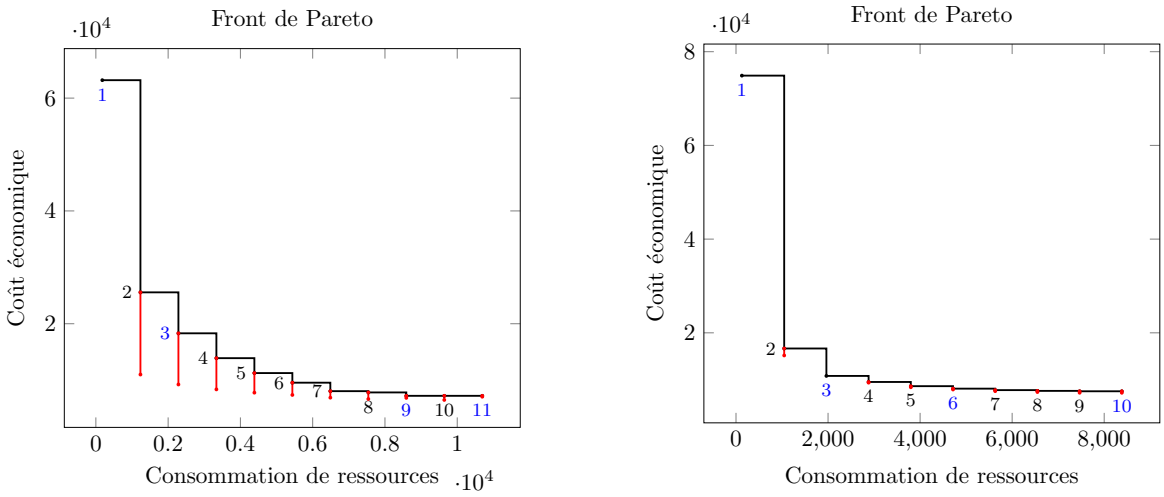


FIGURE 3.7 – Fronts de Pareto obtenus pour $G = 4$ et les deux types de maintenances (parfaite à gauche, imparfaite à droite). Les points sélectionnés sont représentés en bleu.

La méthode de sélection visuelle n'est cependant pas automatisable, et passe donc difficilement à l'échelle. Dans le cas où il n'est pas possible d'effectuer cette sélection visuelle, il est possible de la remplacer par une méthode algorithmique.

1. Sélectionner les deux points extrêmes et les retirer de l'ensemble de Pareto faible
2. Diviser l'ensemble de Pareto faible X en deux parties :
   — $X_1 = \{x_i = (x_i^1, x_i^2) \in X | x_i^2 \leq x_i^1\}$
   — $X_2 = \{x_i = (x_i^1, x_i^2) \in X | x_i^2 > x_i^1\}$
3. Sélectionner le point de X_1 et le point de X_2 ayant le gap absolu le plus faible

Cette méthode de sélection a été évaluée sur l'intégralité du benchmark. Nous présentons les résultats obtenus dans la Table 3.4. Pour chaque groupe d'instance, nous présentons le gap relatif moyen, calculé comme la somme des gaps absolus divisé par la somme des bornes supérieures, dans la colonne 3. Dans la colonne 4, nous explicitons le pourcentage de points sur le front de Pareto pour lesquels un gap relatif inférieur à 10% a pu être obtenu en moins de 15 minutes. Nous présentons dans les colonnes 5 et 6 une mesure d'homogénéité du procédé de sélection respectivement pour l'objectif économique et l'objectif de consommation de ressources. Cette mesure est obtenue en calculant la distance minimale (à gauche) et la distance maximale (à droite) entre deux points consécutifs, puis en les normalisant par la distance entre le point idéal et le nadir. Dans l'ensemble, il apparaît que la méthode nous permet d'obtenir un ensemble de solution satisfaisant avec un gap faible dans le cas de la maintenance imparfaite jusqu'à 8 composants, ou alors jusqu'à deux composants dans le cas de la maintenance parfaite. On constate toutefois que les mesures d'homogénéité sont moyennes, parce que sélectionner le point avec le meilleur gap dans chaque partie ne donne pas nécessairement une sélection homogène. Il reste donc préférable de choisir la méthode de sélection visuelle lorsque c'est possible pour obtenir des ensembles de points plus diversifiés. La méthode de sélection algorithmique reste à ce stade assez rudimentaire et pourrait être

améliorée de diverses manières pour se rapprocher des résultats obtenus par la méthode visuelle.

maint.	nb comp.	min gap moy	% gap < 10%	disp. eco. obj.	disp. res. obj.
parfaite	1	0%	100%	0.023% - 98%	0.032% - 90%
	2	0%	82%	1.1% - 94%	0.063% - 89 %
	4	1%	29%	0% - 75%	8.4% - 78%
	8	11%	13%	0% - 74%	9.4% - 72%
imparfaite	1	0%	98%	0.043% - 98%	0.014% - 88%
	2	0%	85%	0.15% - 92%	0.12% - 90%
	4	1%	79%	0% - 100%	11% - 78%
	8	2%	86%	0% - 100%	8.1% - 89%

TABLE 3.4 – Caractéristiques des fronts de Pareto obtenus

3.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux aspects pratiques de résolution et d'aide à la décision du problème de planification tactique de la maintenance introduit dans le chapitre 2. Étant donné la nature multicritère du problème, nous nous intéressons à l'étude du compromis entre les deux objectifs dans le but de déterminer un ensemble représentatif de bonnes solutions dans un temps raisonnable, afin d'accompagner un preneur de décision. La première approche proposée consiste à fixer quelques valeurs de budget maximum sur la contrainte de ressources et de à résoudre les problèmes mono-objectifs correspondants. La seconde approche s'appuie sur le calcul d'une approximation du front de Pareto par la méthode ε-contrainte combinée à une phase de filtrage permettant d'obtenir un ensemble restreint de solution diversifiées pour accompagner la prise de décision. Les méthodes proposées sont concluantes sur les instances testées. Néanmoins, plusieurs verrous scientifiques restent à lever concernant le temps de calcul des points du front de Pareto et la sélection des points.

Résumé des contributions et perspectives

Principales contributions Dans cette partie, nous avons étudié un problème original de planification tactique de la maintenance faisant intervenir un indicateur de santé des composants d'un équipement et un objectif de minimisation de la consommation de ressources en plus de l'objectif économique classique.

Dans un premier temps, nous étudions la complexité du problème. Nous montrons que dans le cas général le problème est NP-difficile au sens fort et identifions un cas particulier pouvant être résolu en temps pseudo-polynomial par programmation dynamique. Ces similarités nous conduisent à utiliser une approche de programmation linéaire en nombres entiers, très utilisées pour ce type de problèmes.

Dans un second temps, nous nous concentrons sur la résolution en pratique du problème bi-critère. L'objectif est de fournir un outil permettant d'accompagner le preneur de décision dans le choix d'une solution. La première approche proposée consiste à calculer l'optimum de l'objectif économique avec une contrainte sur la consommation maximum de ressources, fournie en entrée du problème. Nous adoptons ensuite une approche bi-critère. Les fronts de Pareto sont approximés à l'aide de la méthode ε-contrainte. L'analyse des résultats obtenus sur des instances générées aléatoirement de taille réaliste nous permet de dériver une méthodologie d'aide à la décision dédiée fournissant un ensemble restreint et diversifié de solutions avec une bonne garantie sur le gap d'optimalité au preneur de décision.

Nous présentons ci-dessous quelques perspectives de recherche autour du problème présenté dans cette partie.

Étude de bornes et heuristiques L'une des principales limitations rencontrées pour la résolution du problème au niveau tactique est le temps de résolution du programme linéaire en nombres entiers, qui ne permet pas de calculer un grand nombre de points du front de Pareto. Les approches heuristiques pour la résolution pourraient permettre de palier cette difficulté, à condition de fournir des solutions de qualité suffisante. Le problème est particulièrement complexe et fait intervenir un grand nombre de contraintes, pour cette raison, développer des heuristiques à performance garantie ne semble pas être une piste prometteuse. Cependant, des heuristiques simples basées sur une approche gloutonne ou métaheuristiques pourraient s'avérer efficaces. En particulier, les algorithmes évolutifs basés sur la population, très largement utilisés dans la littérature de l'optimisation multi-critère, pourraient être intéressants à étudier. Des heuristiques efficaces et l'étude de bornes inférieures, notamment par la résolution de relaxations, pourraient être intégrées éventuellement dans des algorithmes type branch-and-bound.

Amélioration de la procédure d'aide à la décision Nous avons identifié plusieurs facteurs limitants sur la procédure d'aide à la décision qui pourrait faire l'objet de recherches plus approfondies. Le principal problème rencontré est l'inégalité de l'échantillonnage du front de Pareto. Une amélioration de cette procédure, en utilisant par exemple une progression géométrique pour le pas d'échantillonnage permettrait d'avoir plus de points sur la région de compromis, et donc plus de points pertinents parmi lesquels la sélection peut-être effectuée. Enfin, la procédure de sélection algorithmique des points sur le front de Pareto s'avère peu satisfaisante, cette méthode pourrait être améliorée de manière à donner des résultats plus proches de ceux obtenus par la méthode visuelle. Cela pourrait passer notamment par l'intégration d'une mesure d'homogénéité dans la procédure de sélection.

Deuxième partie

Planification opérationnelle de la maintenance

En introduction de ce manuscrit, nous avons introduit une problématique générale autour des problèmes de planification de la maintenance en présence d'un indicateur de santé de l'équipement. Nous avons ensuite séparé les différents aspects de cette problématique en plusieurs niveaux de décision.

Nous nous sommes d'abord concentré, dans la première partie de ce manuscrit, sur la prise de décision au niveau tactique, avec l'étude d'un problème de type lot-sizing pour la planification de la maintenance.

Dans cette partie, nous nous concentrons à présent sur la prise de décision au niveau opérationnel, qui concerne l'ordonnancement à court terme des différentes opérations devant être réalisées sur la machine. Dans le chapitre 4, un problème générique est défini. Divers résultats sont présentés portant sur la complexité du problème et de ses variantes, son approximabilité et sa modélisation par programmation mathématique. Une étude plus approfondie est ensuite proposée pour deux objectifs fondamentaux en ordonnancement théorique : temps de complétion du planning et délai moyen. Un sous-problème particulier, présentant des liens forts avec le bin-packing, est étudié de manière approfondie dans le chapitre 5.

Les travaux présentés dans cette partie ont fait l'objet d'une publication soumise en revue.

Chapitre 4

Ordonnancement conjoint de la production et de la maintenance

Sommaire

Le niveau opérationnel de décision concerne en particulier l'affectation et l'ordonnancement des tâches devant être effectuées sur un horizon de temps court. On s'intéresse ici plus particulièrement à l'ordonnancement conjoint de tâches de production et de maintenance, en présence d'un indicateur de santé de l'équipement. Nous introduisons un nouveau système de notations unifié pour la prise en compte de la dégradation de l'équipement et de la maintenance dans les problèmes d'ordonnancement, basé sur les notations de Graham pour les problèmes d'ordonnancement et les diverses propositions d'extension existantes pour les problèmes d'ordonnancement avec indisponibilités. Ce système de notations est ensuite utilisé pour une revue de l'état de l'art de planification opérationnelle de la maintenance, et tout au long du chapitre pour décrire efficacement les différentes hypothèses des problèmes que nous étudions. Nous étudions par la suite un nouveau problème générique d'ordonnancement conjoint de la production et de la maintenance conditionné par la santé de l'équipement sur un système multicomposant en série. Nous présentons une cartographie de la complexité des différentes variantes de ce problème et proposons une modélisation par programmation linéaire en nombres entiers. Enfin, nous nous intéressons plus spécifiquement à la minimisation du délai maximum et du délai moyen, et présentons divers résultats théoriques pour les différentes variantes.

4.1 Ordonnancement et maintenance

La théorie de l'ordonnancement s'intéresse à la manière d'affecter un ensemble de ressources au cours du temps pour réaliser un ensemble de tâches prédéfinies. Ce type de problème est fréquemment rencontré dans les milieux industriels au niveau opérationnel de décision. Les ressources sont alors par exemple des machines, des unités de productions, des opérateurs... et les tâches correspondent à différentes étapes de la production. Chaque tâche i possède une durée p_i, et étant donné un ordonnancement, on note C_i sa date de fin. La prise de décision concernant la maintenance est souvent étudiée séparément de la prise de décision concernant la production. Il s'agit pourtant de deux problématiques interconnectées, puisque les pannes et les opérations de maintenance affectent le fonctionnement de l'équipement avec un impact direct sur sa disponibilité et son efficacité. Depuis les années 1990, de plus en plus de travaux de recherche s'intéressent à la prise en compte de la maintenance dans les problèmes d'ordonnancement. Nous introduisons dans la sous-section 4.1.1 et la sous-section 4.1.2 un nouveau système de notation et de classification de ces problèmes puis nous proposons dans la sous-section 4.1.3 et la sous-section 4.1.4 un état de l'art de ce pan de la littérature.

4.1.1 Notations de Graham

Les notations de Graham (Graham et al., 1979) font consensus parmi les experts de l'ordonnancement pour décrire de manière compacte les hypothèses des problèmes étudiés. Elle consiste à représenter les différentes hypothèses d'un problème d'ordonnancement en trois champs $\alpha|\beta|\gamma$.

Le premier champ $\alpha = \alpha_1\alpha_2$ correspond aux hypothèses sur l'environnement machine. Le premier paramètre α_1 permet de définir la nature de l'environnement machine. La valeur 1 correspond au cas mono-machine, et P, Q et R correspondent aux différents cas de machines

parallèles. P désigne le cas de machines parallèles identiques, c'est-à-dire ayant la même vitesse d'exécution des tâches. Q désigne le cas des machines parallèles uniformes, dont les vitesses sont proportionnelles les unes par rapport aux autres. Enfin R désigne le cas de machines parallèles indépendantes. On peut également retrouver pour ce paramètre les valeurs F, J et O correspondant à des configurations d'atelier spécifiques, respectivement flow-shop, job-shop et open-shop, que l'on ne détaillera pas ici. Le second paramètre est optionnel et permet de fixer le nombre de machines dans le cas des problèmes à machine parallèles ou d'atelier.

Le second champ $\beta = \beta_1\beta_2\beta_3\beta_4\beta_5$ sert à indiquer les caractéristiques des tâches, tous les paramètres y sont optionnels. Le paramètre β_1 correspond à l'existence de relations de précédence entre les tâches. Ainsi, *prec* implique que certaines tâches ne peuvent pas être exécutées tant que d'autres n'ont pas été terminées. Il est possible de préciser la nature de ces relations de précédence, par exemple *tree* signifient qu'elles forment un arbre, tandis que *chains* signifie qu'elles forment des chaines de précédence. Le paramètre β_2 prend la valeur r_i si les tâches ne peuvent pas être exécutées avant une date de libération. Le paramètre β_3 permet d'indiquer la possibilité de préemption des tâches. La valeur *pmtn* signifie que la préemption est autorisée, et qu'il est alors possible d'interrompre une tâche pour en commencer une autre et la finir plus tard. Le paramètre β_4 prend la valeur d_i si les tâches doivent être finies avant une date limite. Le paramètre β_5 permet d'imposer des contraintes sur le temps d'exécution des tâches p_i.

Enfin, le dernier champ γ permet de représenter l'objectif à optimiser. On peut citer le délai maximum C_{max} et le délai moyen $\sum C_i$, mais également le retard maximum L_{max}, le retard moyen $\sum T_i$ et le nombre de tâches en retard $\sum U_i$. Pour les objectifs de type somme, il est également courant de considérer leurs variantes pondérées $\sum w_iC_i$ $\sum w_iT_i$ et $\sum w_iU_i$.

Ainsi, le problème d'ordonnancement sur deux machines parallèles avec tâches de durée unitaire et deadlines, avec pour objectif la minimisation du délai moyen s'écrit de la manière suivante :

$$P_2|d_j, p_i = 1|\sum C_i$$

Une des grandes forces des notations de Graham est la possibilité de hiérarchiser facilement les différents problèmes de la littérature selon leur complexité et l'ordre de généralisation.

La Figure 4.1 et la Figure 4.2 résument les principales réductions connues concernant l'environnement machine et les objectifs. Ces figures ne sont pas exhaustives et un catalogue plus large peut être retrouvé notamment dans le livre de Pinedo (2012). Lageweg et al. (1982) montrent qu'il est ainsi possible d'étendre les résultats de complexité de manière systématique et algorithmique en s'appuyant sur les notations de Graham et les réductions existantes.

Ainsi, par exemple, si un problème ayant pour objectif $\sum C_i$ est NP-difficile, alors le même problème ayant pour objectif $\sum T_i$ le sera également. Réciproquement, un algorithme permettant de résoudre le problème pour $\sum T_i$, ou des propriétés sur les ordonnancements obtenus, peuvent être transposés à $\sum C_i$.

Les notations présentées ici ne sont bien entendu pas exhaustives et sont étendues au fur et à mesure de l'avancement de la recherche dans le domaine. Le site collaboratif Scheduling Zoo, maintenu par Christoph Dürr, cartographie de manière beaucoup plus complète la littérature des problèmes d'ordonnancement théorique en se basant sur la notation de Graham et les

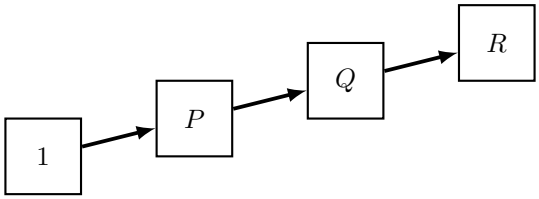


FIGURE 4.1 – Réductions sur l'environnement machine

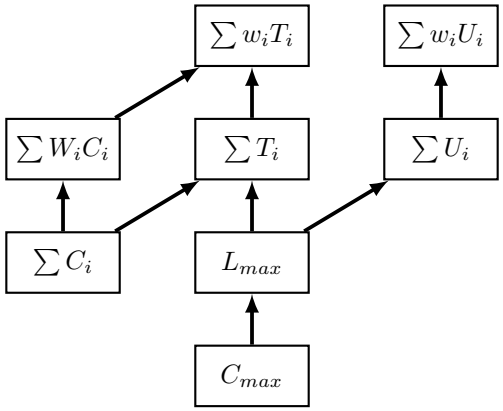


FIGURE 4.2 – Réductions sur les objectifs

différentes réductions connues, en s'appuyant notamment sur l'article de Lageweg et al. (1982).

Ces notations ne conviennent pas pour les problèmes très complexes, faisant intervenir des hypothèses et des contraintes non classiques ou nombreuses ne pouvant pas être représentées de manière compacte dans ce cadre. C'est typiquement le cas de certains problèmes pouvant être rencontrés dans les milieux industriels.

4.1.2 Extension des notations de Graham

Plusieurs propositions ont été faites pour l'intégration de la maintenance dans les notations de Graham, mais il n'existe pas de consensus sur une notation permettant de représenter ces problèmes dans leur diversité. Pour les besoins de leur revue de littérature sur les problèmes d'ordonnancement avec indisponibilité, Ma et al. (2010) ont proposé une synthèse des différentes notations utilisées dans ce domaine. Les auteurs considèrent un troisième champ α_3 permettant de représenter la nature des indisponibilités de la machine. La valeur h_{jk} indique que toutes les machines subissent un nombre quelconque d'indisponibilités. Il est possible de remplacer j par un entier, ce qui permet de préciser le nombre de périodes d'indisponibilités (parfois appelées trous), et également de remplacer k pour indiquer l'indice de la machine qui subit les indisponibilités. Dans le cas à une machine, l'indice k est omis.

Dans le cas d'un nombre quelconque de trous, l'indice j est également omis. Cette notation est complétée dans le paramètre de préemption β_3 pour préciser comment les tâches interagissent avec l'indisponibilité. Dans le cas des tâches sécables, représenté par la valeur $r-a$, une tâche peut être interrompue puis reprise à la fin de l'indisponibilité. Le cas des tâches semi-sécables $sr-a$ est similaire, mais quand une tâche est interrompue, une pénalité est subie. Enfin, le cas non-sécable $nr-a$ signifie que les tâches ne peuvent pas être interrompues par l'indisponibilité. Cette notation présente des ambiguïtés et se limite au cas des indisponibilités, qui sont elles-mêmes un cas particulier de maintenance préventive.

Nous proposons une nouvelle manière d'intégrer la maintenance dans les notations de Graham, qui s'appuie sur la nomenclature habituellement utilisée en maintenance pour classifier les différents problèmes, que nous avons présenté dans le chapitre 1 de manuscrit.

Pour ce faire, nous introduisons un troisième paramètre optionnel α_3 dans le champ α de l'environnement machine, permettant de synthétiser l'ensemble des informations concernant la maintenance, selon la syntaxe suivante $M(\mu_1, \mu_2, \mu_3 \dots \mu_k)$. Le paramètre μ_1 permet de préciser le type de stratégie de maintenance : $prev$ correspond à la maintenance préventive, $pred$ à la maintenance prédictive. Le paramètre μ_2 est optionnel et permet de préciser certaines caractéristiques du système subissant la maintenance. Lorsque la valeur de ce paramètre n'est pas précisé, alors le système considéré est mono-composant. La valeur mcs permet d'indiquer qu'on a un système multicomposant en série. Les valeurs possibles pour les autres paramètres sont introduites tout au long de la revue de littérature.

4.1.3 Maintenance préventive

L'ordonnancement conjoint de la production et de la maintenance préventive a fait l'objet de très nombreux travaux entre les années 1990 et les années 2010. L'approche la plus répandue correspond aux problèmes d'ordonnancement avec indisponibilités, mais d'autres approches telles que des stratégies basées sur l'âge virtuel ont également été investiguées.

Plusieurs revues de littérature existent pour les problèmes d'ordonnancement avec indisponibilités, elles sont énumérées de la plus ancienne à la plus récente. Les revues de Sanlaville et Schmidt (1998) et Schmidt (2000) se concentrent essentiellement sur les problèmes à machines parallèles. Ma et al. (2010) proposent une revue très large des travaux réalisés sur les problèmes à une machine, machines parallèles, flow-shop, open-shop et job-shop avec indisponibilités. Les auteurs résument et présentent dans des tables les différents problèmes, leur complexité et les méthodes exactes et approchées permettant de les résoudre. Enfin, Kaabi et Harrath (2014) couvrent les problèmes d'ordonnancement à machines parallèles, y compris dans les cas uniformes et indépendants, et résument les principaux résultats de complexité et méthodes de résolutions pour chacun de ces problèmes. Dans les deux dernières revues, le cas de l'ordonnancement online est également traités, bien que cela représente une faible partie des travaux dans le domaine.

Il n'existe pas à notre connaissance de revue de littérature plus large concernant la prise en compte de la maintenance préventive au-delà des seules indisponibilités dans les problèmes d'ordonnancement.

Nous excluons de la présente revue la plupart des travaux concernant les problèmes à machine parallèles ou les problèmes d'ordonnancement d'atelier, qui sont moins pertinents dans le contexte de cette thèse, puisque nous nous concentrons sur des systèmes mono-machine.

Ordonnancement avec indisponibilités fixes

En ordonnancement avec indisponibilités, les maintenances préventives sont modélisées comme des périodes où les tâches de production ne peuvent pas être exécutées. Nous reprenons la notation de Ma et al. (2010) dans le champ μ_3 pour les caractériser. La sécabilité est toujours indiquée dans le paramètre de préemption β_3. Dans le but d'alléger les notations, on supposera qu'on est dans le cas des tâches non-sécable $nr-a$ sauf mention explicite du contraire.

On distingue, pour commencer, le cas des indisponibilités fixes, c'est-à-dire que les dates de début et de fin des maintenances sont une donnée du problème connue à l'avance. Ce type d'indisponibilité est signalée dans le champ μ_4 avec la valeur fix. Lee (1996) établit des résultats théoriques de complexité et d'approximabilité pour une très large variété de problèmes d'ordonnancement avec indisponibilités. Il montre que le problème de minimisation du délai maximum avec une seule indisponibilité $1, M(prev, h_1, fix)||C_{max}$ est NP-complet et que l'algorithme LPT (ordonnancement des tâches par ordre décroissant de durée) a un ratio d'approximation de 4/3. Il montre également qu'à partir de deux périodes d'indisponibilité, le problème $1, M(prev, h, fix)||C_{max}$ est NP-complet au sens fort. Breit et al. (2003) montrent qu'à partir de deux périodes d'indisponibilité, le même problème n'admet pas non plus d'algorithme d'approximation à performance garantie à moins que $P = NP$. Plusieurs auteurs se sont également intéressés à une variante du problème où les tâches possèdent une queue, c'est-à-dire un délai q_i entre le moment où la tâche est finie et sa livraison. On cherche alors à maximiser la date de livraison la plus tardive D_{max}. Ce problème, noté $1, M(prev, h, fix)|q_i|D_{max}$, est NP-complet au sens fort dans le cas général. Yuan et al. (2008) montrent que pour un nombre fixe j d'indisponibilité, le problème $1, M(prev, h_j, fix)|q_i|D_{max}$ peut être résolu en temps pseudo-polynomial, mais n'admet pas d'algorithme d'approximation avec une garantie 2^n, où n est le nombre de tâches dès qu'il y a au moins deux indisponibilités. Dans le cas particulier où la longueur totale des tâches est strictement supérieure à la longueur des indisponibilités, ils montrent qu'il n'existe pas de $(2-\epsilon)$-approximation pour tout ϵ positif à moins que $P = NP$. Enfin, un PTAS est proposé pour $1, M(prev, h_1, fix)|q_i|D_{max}$. Kacem (2009) propose une $3/2 - approximation$ et un FPTAS construit à partir d'un programme dynamique pour $1, M(prev, h_1, fix)|q_i|D_{max}$. Kacem et Haouari (2008) adaptent cet FPTAS au cas avec dates de libérations où les queues sont de même longueur $1, M(prev, h_1, fix)|r_i, q_i = q|D_{max}$. Dans le cas général, un algorithme d'approximation avec un ratio $2+\epsilon$ est proposé.

Adiri et al. (1989) se sont intéressés au problème avec une seule machine et une indisponibilité connue à l'avance, pour la minimisation du délai moyen $1, M(prev, h_1, fix)||\sum C_i$. Dans le cas déterministe, ils montrent que le problème est NP-complet. Lee et Liman (1992) montrent que SPT (ordonnancement des tâches par ordre croissant de durée) a une garantie de 9/7 pour ce problème et proposent une preuve de NP-complétude plus simple basée sur EVEN-ODD PARTITION (Garey & Johnson, 1990). Sadfi et al. (2005) améliorent cette garantie à 20/17 en effectuant la procédure de recherche locale $2-OPT$ sur la solution obtenue à l'aide de SPT. Enfin, Y. He et al. (2006) généralisent cette idée dans un PTAS en remplaçant $2-OPT$ par $k-exchange$.

Lee (1996) a étudié la généralisation de ce problème pour le cas du délai moyen pondéré $1, M(prev, h, fix)||\sum w_i C_i$. Il montre que le problème est NP-complet au sens fort en utilisant les réductions classiques d'objectif, et que l'algorithme WSPT (tri par ordre croissant de

durées pondérées) a une erreur relative non bornée y compris si les poids sont égaux aux durées. G. Wang et al. (2005) s'intéressent au cas préemptif $1, M(prev, h, fix)|pmtn|\sum w_iC_i$. Ils montrent que le problème est NP-difficile au sens fort dans le cas général, et dans le cas où les poids sont égaux aux durées, ils montrent que LPT a un ratio d'approximation égal à 2. Ils proposent également une heuristique avec un ratio d'approximation de 2 dans le cas d'une seule indisponibilité. Dans le cas à une seule période d'indisponibilité $1, M(prev, h_1, fix)|pmtn|\sum w_iC_i$, Kacem et Mahjoub (2009) proposent un FPTAS.

Lee (1996) s'est également intéressé aux objectifs $\sum U_i$, L_{max}, également NP-complets au sens fort par généralisation des C_{max} et $\sum C_i$. Pour $\sum U_j$, il montre que l'algorithme de Moore et Hodgson a une erreur absolue de 1. Pour L_{max}, il montre que EDD (ordonnancement des tâches par ordre croissant de dates limites) a une erreur absolue de p_{max}, où p_{max} représente la durée de la tâche la plus longue. Enfin, pour $\sum w_jC_j$, il montre que l'erreur relative est non-bornée, y compris si les poids sont égaux aux durées.

Yazdani et al. (2017) proposent un algorithme de recherche de voisinages pour la minimisation de la somme de l'avance maximum et du retard maximum $1, M(prev, h, fix)|pmtn|ET_{max}$. Ils montrent que ce problème est NP-difficile au sens fort.

Ordonnancement avec indisponibilités périodiques

Dans le cas des problèmes d'ordonnancement avec indisponibilités périodiques, les périodes de maintenance ne sont plus fixées à l'avance en donnée d'entrée du problème et la définition d'un planning de maintenance fait partie du problème à résoudre. La contrainte de périodicité de la maintenance impose alors que deux maintenances successives ne peuvent pas être éloignées temporellement au-delà d'un délai maximum ΔT donné en entrée du problème. Les indisponibilités périodiques sont indiquées par *per* dans le champ μ_4 de notre système de notation.

Qi et al. (1999) s'intéressent au problème de minimisation du délai moyen sur une seule machine avec indisponibilités périodiques $1, M(prev, h, per)||\sum C_i$. Ils montrent que le problème est NP-complet au sens fort et établissent certaines propriétés structurelles sur les solutions optimales : en particulier entre chaque maintenance consécutive, les tâches forment des batchs et au sein de chaque batch, les tâches sont triées selon la règle SPT. Les auteurs proposent un algorithme de branch-and-bound et des heuristiques dédiées. W. J. Chen (2006) propose une heuristique sans garantie de performance, mais très performante en pratique sur les jeux de données de l'article, et un algorithme de branch-and-bound pour le même problème. Qi (2007) établit les ratios d'approximation des algorithmes SPT et EDD pour le même problème ainsi que pour $1, M(prev, h, per)||L_{max}$ en fonction de données d'entrées du problème. Akturk et al. (2003, 2004) étudient indépendamment le même problème, proposent une comparaison de différentes heuristiques, un jeu d'instance et établissent le ratio d'approximation de SPT dans des cas particuliers.

Ji et al. (2007) s'intéressent au cas de la minimisation du délai maximum en présence d'indisponibilités périodiques $1, M(prev, h, per)||C_{max}$, et montrent que LPT est une 2-approximation, et qu'il n'est pas possible d'obtenir de meilleur ratio d'approximation pour ce problème à moins que P=NP. Ce résultat n'est pas contradictoire avec le résultat de Breit et al. (2003) pour le cas des indisponibilités fixes, car il est supposé que les maintenances sont toutes identiques.

Low et al. (2010) étudient une variante de $1, M(prev, h, per)||C_{max}$ sur une seule machine où les indisponibilités sont périodiques, mais où les dates de début des indisponibilités ne sont pas complètement fixées, mais peuvent être déplacées dans une courte fenêtre de temps. Ils montrent que le problème est NP-difficile au sens fort et comparent plusieurs heuristiques de liste sur un jeu d'instances.

Autres approches

D'autres approches de modélisation de la dégradation et de la maintenance en ordonnancement ont été considérées.

Il est courant de modéliser la dégradation de l'équipement dans les problèmes d'ordonnancement par une diminution de la vitesse d'exécution des tâches. Dans les problèmes d'ordonnancement avec "rate-modifying activities", la maintenance permet de réinitialiser la vitesse des machines. Dans notre notation, cette caractéristique est indiquée par le paramètre $M(prev, rmp)$. Ce type de modélisation étant assez éloigné des approches considérées dans cette thèse, nous renvoyons le lecteur vers la revue de littérature de Strusevich, Rustogi et al. (2017) à ce sujet.

Certains travaux en ordonnancement intègrent des politiques de maintenances préventives plus proches des stratégies classiques rencontrées dans la littérature de la planification de la maintenance. Ces problèmes sont en général plus éloignés de la théorie de l'ordonnancement classique et font souvent intervenir de la stochasticité et un nombre plus important de contraintes complexes ou spécifiques typiquement rencontrées dans certains contextes industriels. Pour cette raison, il est souvent difficile d'adapter nos notations à ces problèmes.

Par exemple, Cassady et Kutanoglu (2003) étudient une variante stochastique du problème de minimisation du retard pondéré $1, M(prev, h, per)||w_i T_i$ où la durée d'exploitation avant la prochaine panne suite à une opération de maintenance est décrite par une variable aléatoire. La périodicité des maintenances n'est plus fixée à l'avance, mais doit être déterminée par l'optimisation.

Plusieurs travaux s'intéressent aux stratégies de maintenances basées sur l'âge. Moghaddam et Usher (2011) présentent un problème multicomposant en série où chaque composant a une probabilité de panne dépendante de son âge virtuel. La maintenance préventive permet de réinitialiser l'âge virtuel du composant.

Yang et al. (2019) s'intéressent à un modèle delay-time basé sur l'âge virtuel avec des maintenances imparfaites qui réduisent l'âge virtuel de l'équipement sans le ramener à zéro.

Certains articles se concentrent sur l'optimisation de l'ordonnancement de la maintenance préventive dans des contextes industriels spécifiques. Yao et al. (2004) présentent des modèles de programmation linéaire en nombre entiers pour le cas de la fabrication de semi-conducteurs.

4.1.4 Maintenance prédictive

Les travaux en ordonnancement concernant la maintenance prédictive sont dans l'ensemble plus récents et sont plus rarement abordées avec des techniques classiques de recherche opérationnelle.

L'optimisation des plannings pour des systèmes sujets à la maintenance conditionnelle à fait l'objet de plusieurs travaux de recherche. On peut par exemple citer Grall et al. (2002)

qui décrivent la détérioration d'un système multi-composants à l'aide d'un indicateur de santé permettant de déclencher les opérations de maintenances appropriées au cours d'inspections. Des techniques d'optimisation stochastique sont appliquées pour résoudre le problème étudié. Zhou et al. (2007) modélisent la dégradation à l'aide d'une fonction de fiabilité. La maintenance est déclenchée lorsqu'un seuil de fiabilité est franchi. Les auteurs proposent une résolution analytique du problème considéré.

Avec la montée en puissance des stratégies de maintenance prédictive basées sur les données au cours de ces dernières années, de plus en plus de problèmes d'ordonnancement, théoriques ou appliqués, basés sur des indicateurs pronostiques tels que la santé de l'équipement (notée ehi), sont étudiés.

Kao et al. (2018) étudient l'ordonnancement conjoint de la production et de la maintenance en s'appuyant sur la santé de l'équipement dans le cas de l'industrie des semi-conducteurs. Le problème considéré fait intervenir des contraintes spécifiques au milieu des semi-conducteurs. Il est modélisé et résolu par programmation linéaire en nombres entiers. Penz et al. (2023) adoptent une approche plus proche de l'ordonnancement théorique classique. Les auteurs considèrent le cas de familles de tâches f de production identiques, avec des seuils de santé t_f pour l'exécution de chaque tâche et une usure identique à la durée des tâches $w_f = p_f$. Avec notre système de notations, ce problème peut être exprimé de la manière suivante : $P, M(pred, h_k, ehi)|w_f = p_f, t_f|\sum C_i$ avec $k \in \{1, 2\}$. Les auteurs montrent que certains cas particuliers pour $k = 1$ peuvent être résolus par des algorithmes polynomiaux, mais le problème reste ouvert dans le cas général, tandis que le cas $k = 2$ est NP-difficile. Une approche de programmation linéaire en nombres entiers est proposée pour ce cas difficile.

De nombreux travaux se concentrent sur des cas industriels spécifiques de problèmes d'ordonnancement de la maintenance prédictive. Quelques développements récents dans différents secteurs industriels sont présentés ci-après. Gerum et al. (2019) s'intéressent à l'ordonnancement de la maintenance prédictive dans un contexte ferroviaire et utilisent des méthodes d'optimisation stochastique pour la résolution de leur problème. Wu et al. (2021) étudient les systèmes de chauffage, ventilation et climatisation, et proposent une modélisation faisant notamment intervenir une contrainte de conservation sur la santé de l'équipement se rapprochant des contraintes de conservation présentées dans ce manuscrit. Les auteurs proposent un framework combinant programmation linéaire en nombres entiers et génération de paramètres. de Pater et al. (2022) étudient l'ordonnancement de la maintenance prédictive pour la maintenance de moteurs d'aéronefs, en se basant sur la RUL (Remaining Useful Life) avec un système d'alarmes. Les auteurs proposent un modèle de programmation linéaire en nombres entiers pour l'ordonnancement de la maintenance des composants en état d'alerte. Geng et Wang (2022) s'intéressent quant à eux à l'ordonnancement de la maintenance prédictive sur un réseau électrique et proposent un algorithme dédié pour la résolution de ce problème.

4.2 Définition d'un problème générique d'ordonnancement

Le problème générique étudié dans ce chapitre peut être décrit précisément à l'aide du système de notations de Graham étendues proposé dans la section précédente :

$$1, M(pred, mcs, h, ehi, \mu_5)|\beta|\gamma$$

avec $\mu_5 \in \{perf_g, perf, imperf_g, imperf\}$, $\beta \in \{\emptyset, t_{ig}\}$ et γ pouvant être l'ensemble des objectifs classiques d'ordonnancement évoqués en introduction de cette partie. Nous revenons ici plus en détail sur les caractéristiques de ce problème, et introduisons les notations spécifiques qui seront utilisées tout au long du chapitre.

4.2.1 Données du problème

On considère un problème d'ordonnancement sur une seule machine avec plusieurs composants $g \in \mathcal{G} = \{1 \dots G\}$. La machine possède un niveau de santé par composant évoluant au cours du temps $H_{gt} \in \{0 \dots H_{max}\}$, tel qu'un niveau de santé de 0 correspond à un état défaillant et un niveau de santé de H_{max} correspond à un état neuf (good-as-new).

Un ensemble de tâches $i \in \mathcal{J} = \{J_1 \dots J_n\}$ doit être réalisé par la machine. Ces tâches ont une durée p_i et réduisent le niveau de santé de chaque composant de w_{ig}.

Des opérations de maintenance $m \in \mathcal{M} = \{1 \dots M\}$, de durée π_m permettent de régénérer chaque composant g de la machine d'un montant de santé fixe r_{gm}. Si les modélisations et résultats de complexité proposés s'appliquent dans la plupart des cas à différents types de maintenance, on s'intéressera principalement dans ce chapitre au cas des maintenances parfaites de durées identiques. Dans ce cas, on notera π la durée de ces maintenances.

4.2.2 Contraintes et critères d'optimisation

Nous détaillons à présent les différentes contraintes et les différents objectifs possibles. Parmi ces contraintes, certaines sont optionnelles et correspondent à différentes variantes du problème générique que nous présentons. Une mise en équation de ces contraintes et objectifs est proposée dans la section 4.4.

Ordonnancement mono-machine

Les contraintes classiques des problèmes d'ordonnancement à une machine s'appliquent également au problème proposé ici : l'ensemble des tâches de production doit être planifié sur l'horizon de temps et deux tâches ne peuvent pas être exécutées en même temps. Aucune forme de préemption n'est autorisée, une tâche ayant démarré doit être terminée avant qu'une autre tâche de production ou de maintenance puisse être démarrée.

Évolution de la santé des composants

L'évolution de la santé des composants obéit à plusieurs contraintes.

Tout d'abord, la santé de chaque composant H_{gt} doit rester positive tout au long de l'horizon de temps, une tâche de production i ne peut donc pas être exécutée si la santé d'au moins un des composants est inférieure à l'usure w_{ig} qui serait causée sur ce composant. Cette contrainte traduit le fait que la défaillance d'un des composants de la machine rend impossible son exploitation.

La santé de chaque composant vérifie également une contrainte de conservation du flot, similairement au problème présenté dans la partie I. Après qu'une tâche ait été effectuée, le niveau de santé de chaque composant H_{gt} est égal au niveau de santé $H_{g,t-p_i}$ avant le début de la tâche, auquel est soustrait l'usure w_{ig} causée par la tâche s'il s'agit d'une tâche de

production ou alors auquel est ajoutée la régénération r_{gm} causée par la tâche s'il s'agit d'une tâche de maintenance. Dans le cas de la maintenance, si le niveau de santé devait être amené à dépasser son plafond H_{max}, alors le nouveau niveau de santé est écrêté à ce plafond.

Une autre contrainte, optionnelle, appelée contrainte de seuil, peut être considérée. Un seuil de santé minimal par composant t_{ig} est attribué à chaque tâche. Pour que la solution soit réalisable, la santé de chaque composant avant d'effectuer la tâche doit être supérieure à son seuil. Cette contrainte traduit le fait que certaines tâches ne peuvent pas être réalisées si l'équipement est trop dégradé. On retrouve ce type de contrainte dans le milieu des semi-conducteurs par exemple, où l'exécution de certaines tâches critique sur un équipement dont la santé est en dessous d'un certain seuil d'acceptabilité présente un risque trop important et est par conséquent interdit (Kao et al., 2018).

Opérations de maintenance

De nombreuses hypothèses sont possibles vis-à-vis de la maintenance.

Similairement au problème de la partie I, on distingue le cas de la maintenance répétable, c'est-à-dire que chaque maintenance peut être planifiée autant de fois que nécessaire, du cas de la maintenance unique, pour lequel chaque maintenance peut être planifiée au plus une fois. Ce dernier cas est le plus général : une instance du cas des maintenances répétables peut être construite en dupliquant autant de fois que nécessaire une maintenance unique.

Dans la suite de ce manuscrit, sauf mention explicite du contraire, nous considérerons toujours le cas de la maintenance répétable. On distingue ensuite certains cas particuliers concernant la maintenance. La maintenance parfaite correspond à une opération qui remet un ou plusieurs composant dans l'état de santé maximal H_{max}. La maintenance ciblée, quant à elle, désigne une opération de maintenance qui régénère un unique composant. La maintenance parfaite ciblée est notée $perf_g$, et la maintenance parfaite au sens plus général est simplement notée $perf$. Le cas de la maintenance parfaite et ciblée constitue un cas minimal du point de vue de la complexité, qu'il est ensuite possible de généraliser de très nombreuses manières. On peut par exemple, considérer le cas d'opérations de maintenance imparfaites, noté $imperf$ qui régénèrent une fraction de la santé manquante plutôt qu'un montant fixe, ou encore d'autres fonctions de régénération plus complexes.

Critères d'optimisation

Pour les critères d'optimisation, on s'intéresse ici à deux objectifs classiques d'ordonnancement : le délai maximum C_{max} et le délai moyen $\sum C_i$. Les différents résultats obtenus pour ces critères peuvent être étendus à des critères plus généraux en utilisant les réductions présentées dans la sous-section 4.1.1.

4.2.3 Exemple et visualisation

Dans le cas à un seul composant, il est possible de visualiser une solution sur une variante du diagramme de Gantt. Un exemple pour une instance de $1, M(pred, h, ehi, perf)||C_{max}$ avec $n = 9$ tâches, maintenance parfaite et sans seuils est proposée sur la *Figure* 4.3. Les tâches de production possèdent deux dimensions : une largeur correspondant à leur durée et une hauteur

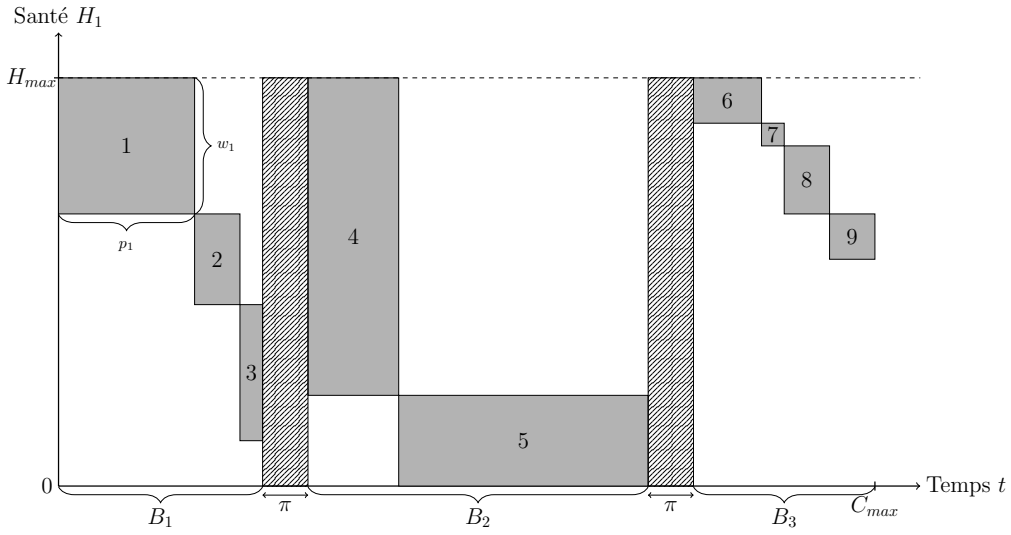


FIGURE 4.3 – Exemple de planning réalisable

correspondant à l'usure qu'elles causent sur l'unique composant. L'axe horizontal correspond au temps, comme dans un diagramme de Gantt classique, cependant, un axe vertical est ajouté pour décrire l'évolution de la santé du composant. Sur l'exemple, initialement, la santé est au maximum, puis la tâche 1 est exécutée, occupant la machine pendant une durée p_1 et causant une usure w_1. La tâche 2 est ensuite exécutée à partir de l'instant $t = p_1$ depuis l'état de santé $H_{1,p_1} = H_{max} - w_1$. Les tâches de maintenances sont représentées par des rectangles hachurés, dont la largeur représente la durée qu'elles occupent.

Dans la solution proposée, les tâches sont placées par ordre d'indice les unes à la suite des autres et des opérations de maintenances sont planifiées dès lors qu'il n'y a plus assez de santé pour la tâche suivante, comme c'est le cas pour la tâche 4 par exemple. En l'absence de contrainte sur le nombre de maintenances, cette heuristique permet toujours de fournir une solution réalisable.

Avec cette visualisation, il est naturel de regrouper les tâches entre deux maintenances consécutives. Ces groupements de tâches sont appelés paquets dans la suite de ce manuscrit. Le paquet de tâches situé entre la maintenance $m-1$ et m (à l'exception du premier et du dernier paquet, qui ne sont respectivement pas précédé ou suivi d'une maintenance) est noté B_m.

L'un des intérêts de cette visualisation est qu'elle fait directement ressortir le lien entre notre problème d'ordonnancement et le problème de bin-packing classique. Il s'agit de regrouper efficacement les tâches entre les maintenances consécutives, de sorte à optimiser les critères considérés.

Le lien entre ces deux problèmes est approfondi dans le chapitre 5 pour le cas de la minimisation du délai maximum.

4.3 Étude de complexité

Dans cette section, un panorama de la complexité du problème en fonction des différentes hypothèses est proposé. La méthodologie consiste à établir la NP-complétude pour les problèmes les plus minimaux possibles, puis d'établir une cartographie des réductions entre les différentes variantes et hypothèses afin d'apporter une vue d'ensemble sur la complexité du problème.

4.3.1 Problèmes NP-complets minimaux

On supposera donc ici que la machine n'a qu'un seul composant et que la maintenance est parfaite. Dans un premier temps la NP-complétude au sens fort du problème d'existence d'une solution avec un nombre de maintenances contraint est démontré, puis nous montrons en corollaire que le problème de minimisation du délai maximum C_{max} est NP-difficile au sens fort. Dans un second temps, nous montrons que le problème de minimisation du délai moyen $\sum C_i$ est également NP-difficile au sens fort par réduction d'un problème de l'état de l'art.

Existence d'une solution

Nous nous intéressons dans un premier temps à l'existence d'une solution. Dans la section précédente, nous avons constaté qu'en l'absence de contrainte sur le nombre de maintenances, une méthode gloutonne permet de construire un planning réalisable. Toutefois, lorsque l'on considère un nombre limité de maintenances, le problème devient NP-complet au sens fort. Nous considérons le problème de décision $\mathcal{D} = 1, M(pred, h_k, ehi, perf)||\emptyset$, où la machine est monocomposant, les maintenances sont parfaites et le nombre de maintenances autorisé est borné par un entier M. La question est de savoir s'il existe une solution réalisable. Notons que la durée des tâches n'intervient pas dans la faisabilité d'un planning, il n'est donc pas nécessaire de les préciser dans une instance de $\mathcal{D}$.

Proposition 4.3.1. *Le problème $\mathcal{D} = 1, M(pred, h_k, ehi, perf)||\emptyset$ est NP-complet au sens fort.*

Démonstration. Le problème $\mathcal{D}$ est bien un problème de décision, et, étant donné une solution, il suffit de vérifier que l'ensemble des contraintes du problème présentées dans la sous-section 4.2.2 sont vérifiées pour s'assurer que la solution est réalisable. Cette vérification s'effectue en temps polynomial, d'où l'appartenance de $\mathcal{D}$ à NP.

Montrons qu'il est NP-difficile au sens fort en réduisant 3-PARTITION.

Définition 4.3.2. *3-PARTITION (Garey & Johnson, 1990)*

***Instance** : Un ensemble fini d'items A, $|A| = 3l$, $l \in \mathbb{N}$, une borne $K \in \mathbb{N}$ et une taille $s(a) \in \mathbb{N}$ pour tout $a \in A$ tels que $\forall a \in A, K/4 < s(a) < K/2$ et $\sum_{a \in A} s(a) = lK$.*

***Question** : Existe-t-il une partition de A en l ensembles disjoints $A_1 \dots A_l$ tels que pour tout $1 \leq i \leq l$, $\sum_{a \in A_i} s(a) = K$?*

La démonstration consiste à regrouper chaque sous ensemble de la partition entre deux maintenances. On va ainsi construire la partition en plaçant les tâches entre les maintenances, et le nombre de partitions sera borné par le nombre de maintenances.

Soit I, une instance de 3-PARTITION avec les notations de la définition, notons f, l'application transformant une instance de 3-PARTITION en une instance de $\mathcal{D}$ de la manière suivante :

— $\mathcal{J} = \{J_1 \dots J_{3l}\}$ avec $\forall i \in \mathcal{J},\ w_i = s(i)$
— $H_{max} = K$
— $M = l - 1$

Soit $Y_{\mathcal{D}}$ (resp. Y_{3P}) ensemble des instances "oui" de $\mathcal{D}$ (resp. 3-PARTITION). On utilise la notion de paquets de tâches mentionnée en introduction du problème. On note ainsi B_m ensemble des tâches planifiées entre la $(m-1)$-ème et la m-ième maintenance pour une solution de $\mathcal{D}$ (les maintenances 0 et M n'étant pas des maintenances, mais correspondant au début et à la fin du planning). Montrons que $I \in Y_{3P} \iff f(I) \in Y_{\mathcal{D}}$:

⇒ Étant donné une instance I de Y_{3P} et une solution correspondante, construisons la solution de $\mathcal{D}$ telle que les tâches du paquet B_m correspondent aux items du sous-ensemble A_m. On a, par hypothèse, que pour tout $1 \leq i \leq l$, $\sum_{a \in A_i} s(a) = K$, ce qui implique, en appliquant f et la construction proposée que pour tout $1 \leq m \leq l$, $\sum_{i \in B_m} w_i = H_{max}$. De plus, toutes les tâches sont planifiées, et on a l paquets, soit exactement $l - 1 = M$ maintenances. On a donc $f(I) \in Y_{\mathcal{D}}$.

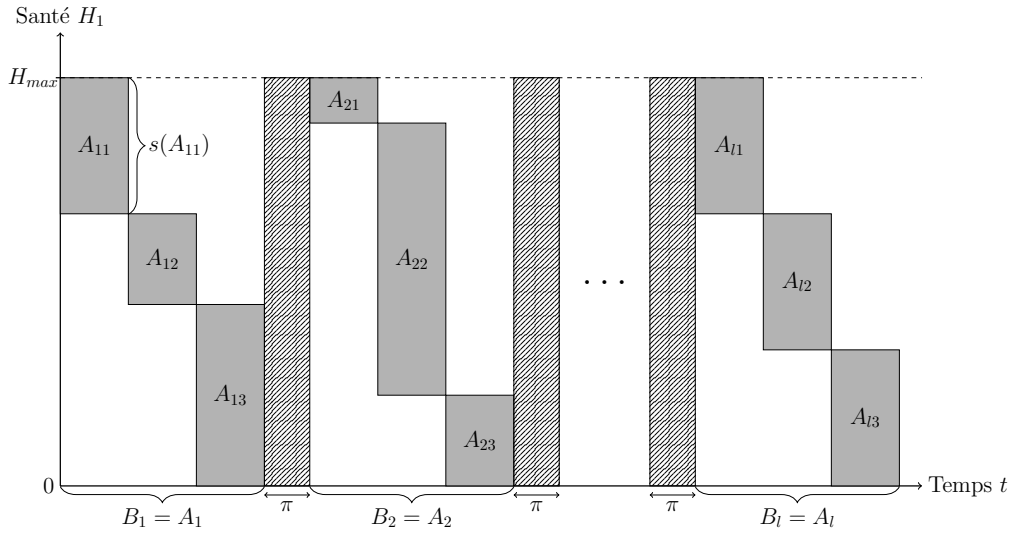


FIGURE 4.4 – Réduction de 3-PARTITION dans $\mathcal{D}$

⇐ A contrario, si I est une instance de 3-PARTITION telle que $f(I) \in Y_{\mathcal{D}}$. Alors pour tout $1 \leq m \leq l$, $\sum_{i \in B_m} w_i \leq H_{max}$. En construisant la solution de 3-PARTITION telle que les items correspondant aux tâches d'un même paquet sont rangés dans le même sous-ensemble. La nouvelle solution vérifie par définition de la transformation, pour tout $1 \leq i \leq l$, $\sum_{a \in A_i} s(a) \leq K$. Or, I est une instance de 3-PARTITION et vérifie donc par définition $\sum_{a \in A} s(a) = lK$. Cette égalité n'est possible que si chacune des l inégalités sont saturées, d'où, pour tout $1 \leq i \leq l$, $\sum_{a \in A_i} s(a) = K$. On a finalement bien $I \in Y_{3P}$.

La réduction proposée vérifie l'ensemble des hypothèses nécessaires d'une réduction pseudo-polynomiale depuis un problème NP-complet au sens fort, ce qui permet de conclure que $\mathcal{D}$ est NP-complet au sens fort.

□

Minimisation du délai maximum

Le problème de minimisation du délai maximum, noté $\mathcal{P}$, est directement lié au problème d'existence $\mathcal{D} = 1, M(pred, h_k, ehi, perf)||\emptyset$ étudié précédemment.

En effet, comme il s'agit d'un problème mono-machine, l'agencement des tâches de production n'a aucun impact sur le délai maximum. Seul le temps de maintenance est compressible, qui peut être minimisé en minimisant le nombre de maintenances. Le problème de décision sous-jacent est donc le problème d'existence $\mathcal{D}$, comme montré ci-dessous.

Corollaire 4.3.3. *Le problème de minimisation du délai maximum C_{max} avec maintenance parfaite réutilisable à l'infini $\mathcal{P} = 1, M(pred, h, ehi, perf)||C_{max}$ est NP-difficile au sens fort.*

Démonstration. Supposons qu'on cherche un planning réalisable avec au plus M maintenances. Il suffit de poser :

$$C_{max} \leq C = \sum_{i \in \mathcal{J}} p_i + \pi(M-1)$$

pour se ramener au problème de décision associé à $\mathcal{P} = 1, M(pred, h, ehi, perf)||C_{max}$. L'ensemble des instances de $\mathcal{D}$ est donc inclus dans l'ensemble des instances du problème de décision associé à $\mathcal{P}$, d'où $\mathcal{P}$ NP-difficile au sens fort. Réciproquement, chercher un planning tel que $C_{max} \leq C$ revient à chercher un planning avec un nombre de maintenances maximum $M = \lfloor \frac{C - \sum_{i \in \mathcal{J}} p_i}{\pi} \rfloor$. □

Minimisation du délai moyen

Nous nous intéressons à présent au problème de minimisation du délai moyen sur une machine mono-composant lorsque la maintenance est parfaite $1, M(pred, h, ehi, perf)||\sum C_i$. Qi et al. (1999) montrent que le problème d'ordonnancement sur une seule machine avec indisponibilités flexibles pour la minimisation du délai moyen $1, M(prev, h, per)||\sum C_i$ est NP-difficile au sens fort par réduction de 3-PARTITION. Dans ce problème, les auteurs considèrent que la fenêtre de temps où des tâches de production peuvent être planifiées entre deux opérations de maintenance consécutives doit être inférieure à un délai ΔT, donné en entrée du problème. Pour le reste, il s'agit d'un problème d'ordonnancement classique de la production, dont l'objectif est la minimisation du délai moyen $\sum C_i$. Il est possible de transformer une instance de ce problème en une instance de $1, M(pred, h, ehi, perf)||\sum C_i$ en posant une usure proportionnelle à la durée des tâches, soit $w_i = \frac{p_i H_{max}}{\Delta T}$. Ainsi, $1, M(prev, h, per)||\sum C_i$ est un cas particulier de $1, M(pred, h, ehi, perf)||\sum C_i$, qui est donc NP-difficile au sens fort par transitivité.

4.3.2 Réductions

Nous cherchons à présent à hiérarchiser les différentes hypothèses du problème en termes de complexité, en s'appuyant sur le système de notations de Graham étendu évoqué précé-

demment. Du côté de l'environnement machine (hormis la maintenance) et des objectifs, nous n'introduisons pas de nouvelles hypothèses spécifiques au problème. Les réductions classiques de la littérature ont été rappelées sur la Figure 4.1 et la Figure 4.2. Nous présentons ci-après les réductions concernant la maintenance et les caractéristiques des tâches spécifiques au problème.

Concernant la maintenance, nous avons introduit de nouvelles hypothèses concernant la nature de la maintenance et sur l'aspect multi-composants en séries. On note que la maintenance parfaite est un cas particulier de maintenance imparfaite, et que la maintenance ciblée est également un cas particulier de maintenance. Nous avons également vu que la limitation du nombre de maintenances autorisé impacte également la complexité du problème. La Figure 4.5 synthétise les réductions entre ces différentes hypothèses.

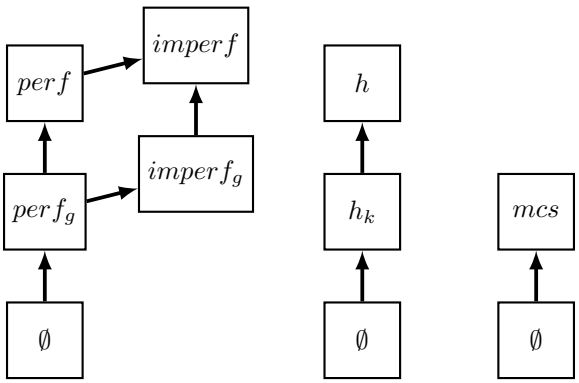


FIGURE 4.5 – Réductions sur les caractéristiques des maintenances

Concernant les caractéristiques des tâches, nous n'introduisons qu'une seule nouvelle hypothèse concernant les seuils de santé pour l'exécution des tâches. La réduction correspondante est représentée sur la Figure 4.6.

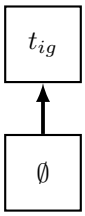


FIGURE 4.6 – Réductions sur les caractéristiques des tâches

Nous avons montré précédemment que les trois problèmes $1, M(pred, h_k, ehi, perf)||\emptyset$; $1, M(pred, h, ehi, perf)||C_{max}$ et $1, M(pred, h, ehi, perf)||\sum C_i$ sont NP-difficiles au sens fort. Ces problèmes correspondent aux hypothèses parmi les plus simples vis-à-vis de la hiérarchie

des différentes variantes en termes de complexité. Les réductions présentées précédemment permettent de dériver de nombreux résultats de NP-complétude pour les différentes variantes du problème. Cela nous conduit à nous tourner davantage vers des approches heuristiques ou de programmation mathématique pour la résolution pratique.

4.4 Programmation mathématique

Les approches de programmation mathématique, en particulier la programmation linéaire en nombres entiers sont très largement utilisées pour modéliser les problèmes d'ordonnancement. Elles permettent de formaliser et de mettre en équation les contraintes des problèmes et peuvent être également utilisées pour la résolution. On rencontre néanmoins des difficultés en pratique pour la linéarisation des contraintes de séquence, ce qui limite son utilisation pour la résolution. Il existe une large variété de modèles permettant de formuler les différentes contraintes habituelles des problèmes d'ordonnancement. Ces modèles ont été recensés par Keha et al. (2009) pour le cas des problèmes à une seule machine.

Dans le cadre de notre problème, la modélisation de la conservation de la santé s'avère particulièrement délicate. Sauf dans le cas de la formulation indexée par le temps, des techniques de linéarisation faisant intervenir un très grand nombre de variables et de contraintes sont nécessaires. Cette difficulté est due au fait que si une maintenance devait amener la santé d'un composant au-dessus du plafond de santé H_{max} (en particulier dans le cas de la maintenance parfaite), la nouvelle santé serait alors écrêtée à H_{max}. Cette contrainte est particulièrement difficile à linéariser. On se concentre donc uniquement sur une formulation indexée par le temps.

Variables de décision Considérons un horizon de temps $\mathcal{T} = \{0 \dots T\}$ où T est une borne supérieure de la durée d'un planning. On introduit deux types de variables de décision :

$$x_{it} = \begin{cases} 1 & \text{si la tâche } i \in \mathcal{J} \cup \mathcal{M} \text{ commence à l'instant } t \\ 0 & \text{sinon.} \end{cases}$$

Cet ensemble de variables représente la décision sur la position des tâches de production et de maintenance dans l'ordonnancement au cours du temps.

$$H_{gt} \in \mathbb{N}$$

Cet ensemble de variables représente l'évolution de la santé des composants au cours du temps.

Contraintes On peut distinguer deux types de contraintes, les contraintes génériques des problèmes d'ordonnancement à une seule machine et les contraintes spécifiques au problème.

Les contraintes génériques doivent être légèrement adaptées pour prendre en compte les tâches de maintenance, qui ont un comportement légèrement différent. Si on fait l'hypothèse que chaque opération de maintenance peut être planifiée autant de fois que nécessaire, ces contraintes s'expriment de la manière suivante.

Une première contrainte (4.1) impose que les tâches de maintenance comme de production ne peuvent pas se superposer dans le temps.

$$\forall t \in \mathcal{T}, \quad \sum_{i \in \mathcal{J}} \sum_{\tau=max(0,t-p_i+1)}^{t} x_{i\tau} + \sum_{m \in \mathcal{M}} \sum_{\tau=max(0,t-\pi_m+1)}^{t} x_{m\tau} \leq 1 \tag{4.1}$$

Une seconde contrainte (4.2) impose que chaque tâche de production apparaisse exactement une seule fois.

$$\forall i \in \mathcal{J}, \quad \sum_{t=0}^{T-p_i+1} x_{it} = 1 \tag{4.2}$$

Les contraintes spécifiques au problème concernent l'évolution de la santé des composants. Une première contrainte (4.3) impose que la santé de chaque composant ne peut excéder son plafond.

$$\forall t \in \mathcal{T}, \quad \forall g \in \mathcal{G}, \quad H_{gt} \leq H_{max} \tag{4.3}$$

La conservation de la santé des composants est exprimée avec la contrainte suivante :

$$\forall t \in \{0 \dots T-1\}, \quad H_{gt+1} \leq H_{gt} + \sum_{m \in \mathcal{M}} r_{gm} x_{mt} - \sum_{i \in \mathcal{J}} w_{ig} x_{it} \tag{4.4}$$

Similairement au modèle de programmation linéaire en nombre entier présenté dans le chapitre 2, on a une inégalité au lieu d'une égalité en raison du plafond de santé. Si la maintenance ramène la santé au-dessus du plafond, alors la santé est écrêtée et une partie de la régénération est perdue.

Objectifs Avec les variables de décision du modèle, on peut exprimer le délai d'une tâche de la manière suivante :

$$C_i = \sum_{t=0}^{T-p_i+1} (t + p_i) x_{it} \tag{4.5}$$

Le délai moyen $\sum C_i$ s'exprime alors comme la somme de ces termes pour chacune des tâches de production.

Le délai total du planning, quant à lui, s'exprime de la manière suivante :

$$C_{max} = \sum_{i \in \mathcal{J}} p_i + \sum_{t=0}^{T} \sum_{m \in \mathcal{M}} x_{mt} \tag{4.6}$$

Notons que le terme $\sum_{i \in \mathcal{J}} p_i$ ne fait pas intervenir de variables de décision, le délai maximum de l'ordonnancement dépend uniquement de la décision de maintenance.

Il est possible d'exprimer les autres objectifs classiques d'ordonnancement comme le retard pondéré ou le nombre de tâches en retard sans ajouter de contraintes au modèle, de manière identique au problème d'ordonnancement mono-machine classique. La présence de la maintenance et de la santé des équipement ne changent pas leur expression dans la formulation indexée par le temps. Les expressions de ces objectifs sont présentées dans la revue de Keha et al. (2009).

Variantes Le modèle présenté est très général et plusieurs sous-problèmes peuvent être retrouvés en choisissant des valeurs spécifiques pour les paramètres. En particulier, le cas de la maintenance parfaite est retrouvé en posant une maintenance par composant tel que $r_{gm} = H_{max}$ si $g = m$ et 0 sinon.

Le cas où chaque opération de maintenance peut être planifiée au plus une fois peut être modélisé en ajoutant la contrainte suivante :

$$\forall m \in \mathcal{M}, \quad \sum_{t=0}^{T-\pi_m+1} x_{mt} \leq 1 \tag{4.7}$$

La contrainte de seuil peut être également modélisée en ajoutant la contrainte suivante.

$$\forall t \in \mathcal{T}, \quad \forall g \in \mathcal{G}, \quad \sum_{i \in \mathcal{J}} t_{ig} x_{it} \leq H_{gt} \tag{4.8}$$

Autres formulations La formulation indexée par le temps permet de formaliser le problème, mais est peu efficace en pratique pour la résolution des problèmes d'ordonnancement. Les autres formulations classiques ont été investiguées pour notre problème, mais le comportement non-linéaire de la santé lors des opérations de maintenance rend très difficile, selon la formulation choisie, la modélisation de la santé de l'équipement ou alors de la date de fin des tâches. Dans le cas de la minimisation de C_{max}, il est possible de contourner cette difficulté en exploitant la relation du problème avec les problèmes de bin-packing. Des formulations alternatives sont ainsi présentées dans le chapitre 5.

4.5 Minimisation du délai maximum

Le problème de minimisation du délai maximum est très lié au problème de bin-packing. Dans cette section, nous présentons ce lien pour le problème $1, M(pred, h_k, ehi, perf)||C_{max}$, et proposons quelques résultats d'approximabilité. Une étude plus approfondie en présence de la contrainte de seuil est proposée dans le chapitre 5.

4.5.1 Inapproximabilité

Le problème de minimisation du délai maximum avec maintenance parfaite et un seul composant est rigoureusement équivalent au problème de bin-packing, moyennant la translation de l'objectif utilisée pour la démonstration du Corollaire 4.3.3 dans la section 4.3.1.

Pour la minimisation du nombre de maintenances, la durée des tâches de production et de maintenance n'ont aucun impact sur les solutions et sur la valeur de l'objectif et peuvent donc être ignorées. La transformation d'une instance de notre problème d'ordonnancement en un problème de bin-packing se fait de la manière suivante : pour chaque tâche, l'item correspondant avec un poids égal à l'usure de la tâche est créé et la capacité des boîtes est fixée à H_{max}.

Il est donc possible d'adapter la preuve d'inapproximabilité du bin-packing, pouvant être retrouvée par exemple dans l'ouvrage de référence de Martello et Toth (1990) sur le problème de sac à dos et ses dérivés.

Par souci de simplicité, nous présentons ce résultat directement pour la minimisation du nombre de maintenances $1, M(pred, h_k, ehi, perf)||k$. Comme les paquets correspondent aux boîtes, et qu'il y a une maintenance de moins que le nombre de boîtes, le ratio d'inapproximabilité est décalé.

Proposition 4.5.1. *Le problème de minimisation du nombre de maintenances, noté $\mathcal{P}_k = 1, M(pred, h_k, ehi, perf)||k$, n'est pas $(2-\epsilon)$-approximable pour tout ϵ strictement positif à moins que $P = NP$.*

La preuve consiste à montrer qu'une $(2-\epsilon)$-approximation du problème pourrait être utilisée pour résoudre le problème PARTITION, NP-complet (Garey & Johnson, 1990), en temps polynomial, ce qui est contradictoire à moins que $P = NP$.

4.5.2 Heuristiques à performances garanties de bin-packing

Il est possible de transposer les différents algorithmes utilisés pour résoudre le bin-packing classique, et en particulier les algorithmes d'approximation en conservant les mêmes garanties de performance.

Comme la durée des tâches n'a pas d'impact sur la nature de la solution optimale dans le cas de la minimisation de C_{max}. Si un algorithme de bin-packing fournit une solution contenant p boîtes, alors cette solution peut être transposée en une solution de notre problème d'ordonnancement avec $p-1$ maintenances en faisant la transformation inverse.

On peut ainsi exprimer le ratio d'approximation d'un algorithme de bin-packing pour $1, M(pred, h_k, ehi, perf)||C_{max}$ en fonction du ratio d'approximation pour le problème bin-packing.

Proposition 4.5.2. *Si $\sum_{i\in\mathcal{J}} p_i \geq \pi$, alors le ratio d'approximation de tout heuristique à performance garantie du bin-packing est conservé pour le problème $1, M(pred, h_k, ehi, perf)||C_{max}$.*

Démonstration. Notons I une instance de notre problème, $f(I)$ l'instance correspondante de bin-packing, $\mathcal{A}_{BP}$ un algorithme d'approximation de bin-packing, $\mathcal{A}$ l'algorithme correspondant pour notre problème. On note également OPT et OPT_{BP} les solutions optimales respectives, et ρ_{BP} le ratio d'approximation de $\mathcal{A}_{BP}$, alors :

$$
\begin{aligned}
OPT(I) &= \sum_{i\in\mathcal{J}} p_i + \pi\left(OPT_{BP}(f(I)) - 1\right) \\
\mathcal{A}(I) &= \sum_{i\in\mathcal{J}} p_i + \pi\left(\mathcal{A}_{BP}(f(I)) - 1\right) \\
&\leq \sum_{i\in\mathcal{J}} p_i + \pi\left(\rho_{BP} OPT_{BP}(f(I)) - 1\right) \\
&= \sum_{i\in\mathcal{J}} p_i + \pi\left(OPT_{BP}(f(I)) - 1\right) + \pi(\rho_{BP} - 1)\left(OPT_{BP}(f(I))\right) \\
&= OPT(I) + (\rho_{BP} - 1)\left(\pi + \pi(OPT_{BP}(f(I)) - 1)\right)
\end{aligned}
$$

Comme $\sum_{i\in\mathcal{J}} p_i \geq \pi$, c'est-à-dire que la durée d'une maintenance est plus courte que la durée de toutes les tâches.

Alors :

$$\mathcal{A}(I) \leq OPT(I) + (\rho_{BP} - 1)\left(\sum_{i \in \mathcal{J}} p_i + \pi(OPT_{BP}(f(I)) - 1)\right) = \rho_{BP} OPT(I)$$

Sous l'hypothèse que la durée d'une maintenance est plus courte que la durée totale des tâches, on retrouve bien le ratio d'approximation des algorithmes de bin-packing.

□

4.5.3 Algorithmes online dans le cas multi-composant

Dans cette section, nous nous intéressons aux algorithmes online pour le problème multi-composant. Nous montrons que sous certaines hypothèses concernant l'échantillonnage de la santé, le ratio compétitif de tout algorithme online admet une borne inférieure dépendant du nombre de composants. Dans la continuité des sections précédentes, nous étudions le problème de minimisation du nombre de maintenances, équivalent à la minimisation du délai maximum. Dans le système de notations de Graham étendu, le problème s'écrit : $1, M(pred, mcs, h_k, ehi, perf)||k$.

Proposition 4.5.3. *Si, $G < H_{max}/2$, le ratio de compétitivité de tout algorithme online pour $1, M(pred, mcs, h_k, ehi, perf)||k$ est au moins $G - 1$, où G est le nombre de composants.*

Démonstration. La preuve consiste à considérer deux ensembles de tâches distinctes : un premier ensemble de tâches $\mathcal{N}_1$ qui consomment la quasi-totalité d'un composant et un second ensemble de tâches $\mathcal{N}_2$ qui consomment une toute petite fraction de la santé de tous les composants. Ainsi les tâches de $\mathcal{N}_1$ peuvent toutes êtres regroupées dans le même groupe, et de même pour les tâches de $\mathcal{N}_2$. En revanche, il n'existe qu'une seule manière de regrouper une tâche de $\mathcal{N}_1$ et une tâche de $\mathcal{N}_2$ dans un même paquet, et cela exclut toutes les autres tâches. Ainsi si l'ordre de tâches est imposé, il est possible d'obtenir une solution $G - 1$ fois pire que l'optimum, comme présenté ci-après.

Soit une instance avec l'ensemble de tâches suivant $\mathcal{J} = \mathcal{N}_1 \cup \mathcal{N}_2$, où :

— $\mathcal{N}_1$ contient G tâches, telles que la g-ième tâche, notée $\mathcal{N}_1(g)$, consomme $H_{max} - 1$ de la santé du composant g et 0 des autres.

— $\mathcal{N}_2$ contient G tâches, telles que la g-ième tâche, notée $\mathcal{N}_2(g)$ consomme 1 de la santé du composant g et 2 des autres.

Supposons que les tâches arrivent une par une dans l'ordre suivant :

$$\mathcal{N}_1(1), \mathcal{N}_2(1), \mathcal{N}_1(2), \mathcal{N}_2(2) ... \mathcal{N}_1(G), \mathcal{N}_2(G)$$

Supposons que la maintenance régénère tous les composants à H_{max}, en la notant M dans le planning, puisque les tâches sont libérées une par une, tout algorithme online donnera la solution suivante :

$$\mathcal{N}_1(1), \mathcal{N}_2(1), M, \mathcal{N}_1(2), \mathcal{N}_2(2), M, ..., M, \mathcal{N}_1(G), \mathcal{N}_2(G)$$

Ce planning contient $G - 1$ maintenances, alors que le planning suivant, contenant une seule maintenance, est réalisable :

$$\mathcal{N}_1(1), \mathcal{N}_1(2) ... \mathcal{N}_1(G), M, \mathcal{N}_2(1), \mathcal{N}_2(2) ... \mathcal{N}_2(G)$$

Ainsi, pour cette instance, la solution obtenue par tout algorithme online fait intervenir $G-1$ maintenances, tandis que l'optimum offline fait intervenir une seule maintenance. Le ratio de compétitivité de tout algorithme online est donc au moins $G-1$.

□

Pour plus de $H_{max}/2$ composants, il n'y a plus assez de santé pour permettre de planifier toutes les maintenances du groupe $\mathcal{N}_2$ sans maintenance supplémentaire. Cependant en autorisant des valeurs rationnelles ou avec un meilleur échantillonnage de la santé, il est possible d'adapter la preuve pour que le résultat reste valide. Ce résultat signifie que quand le nombre de composants devient important, il n'existe pas de bonne garantie de performance pour les algorithmes online.

4.6 Minimisation du délai moyen

La minimisation du délai moyen $\sum C_i$ est un autre objectif classique en ordonnancement. Dans la section 4.3, nous avons montré que le problème $1, M(pred, h, ehi, perf) || \sum C_i$ est NP-difficile au sens fort, et en utilisant les différentes réductions connues, un très large éventail de problèmes de minimisation du délai moyen sont également NP-difficiles au sens fort.

Dans cette section, nous étudions la structure des solutions optimales du cas minimal $1, M(pred, h, ehi, perf) || \sum C_i$. Nous revenons en particulier sur le problème d'ordonnancement avec indisponibilité étudié par Qi et al. (1999), noté $1, M(prev, h, per) || \sum C_i$, que notre problème généralise et montrons que certaines propriétés sont conservées.

On s'intéresse dans un second temps au cas particulier des familles de tâches, résolu par programmation dynamique, ce qui nous permet de dériver une suite d'algorithmes convergeant vers un algorithme de résolution du cas général.

4.6.1 Réorganisation des termes de $\sum C_i$

Commençons par établir un résultat sur l'expression de l'objectif utile pour les démonstrations à venir. Il est possible de réorganiser les termes de $\sum C_i$ en regroupant les paquets. On suppose que les tâches sont ordonnées de 1 à n dans une solution et regroupées en $M+1$ paquets $\{B_1 \dots B_{M+1}\}$ séparés par M maintenances.

Le principe de cette décomposition est illustrée pour une solution à trois paquets sur la Figure 4.7. La hauteur de chaque barre correspond au délai maximum C_m de chacune des tâches. La surface totale de la figure correspond alors à la valeur du délai moyen. Les parties hachurées correspondent à la durée de maintenance.

On introduit les notations suivantes :

— γ_m délai moyen pour le sous-ordonnancement des taches du paquet B_m (en considérant que la date de début du paquet est 0).
— $p(B_m) = \sum_{i \in B_m} p_i$ temps d'exécution de l'ensemble de tâches de B_m.
— p_{jm} durée de la j-ième tâche du m-ième paquet.

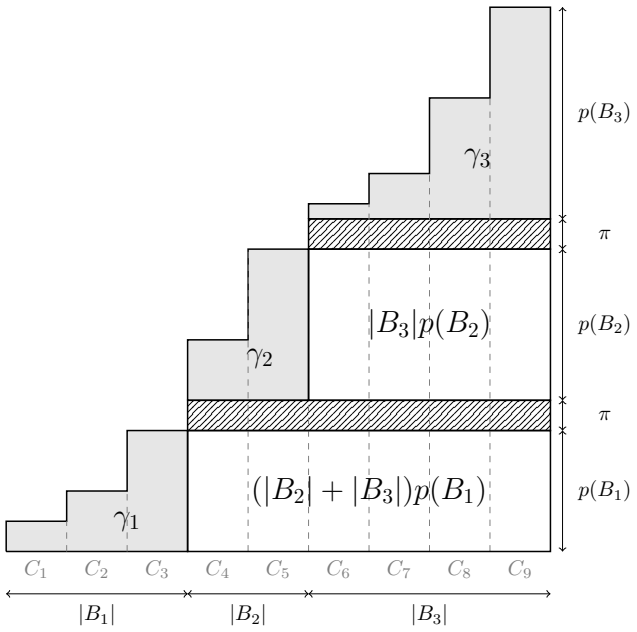


FIGURE 4.7 – Illustration de la réorganisation des termes de la somme

Il apparaît sur la figure alors qu'il est effectivement possible de décomposer la somme avec les termes γ_m correspondant aux parties grisées de la figure, et des rectangles, dont la hauteur est égale au délai des tâches, plus la durée de maintenance, et la largeur égale aux nombres de tâches dans les paquets restants.

On a alors :

Lemme 4.6.1. $\sum_{i\in\mathcal{J}} C_i = \sum_{m=1}^{M+1} \gamma_m + \sum_{m=1}^{M} \left(\sum_{l=m+1}^{M+1} |B_l|\right) (p(B_m)+\pi)$

Démonstration. L'idée de la démonstration repose sur le réarrangement des termes de $\sum C_i$. On peut constater que dans la somme finale, on compte n fois la première tâche, $n-1$ fois la deuxième et ainsi de suite jusqu'à une fois la dernière tâche. De plus il faut également compter les termes de maintenance, ce qui correspond, pour chaque tâche, à la durée totale de maintenance qui la précède, c'est-à-dire le produit de la durée de la maintenance par le nombre de paquets avant elle. Formellement :

$$\sum_{i\in\mathcal{J}} C_i = \underbrace{\sum_{i=1}^{n} (n-i+1)p_i}_{f_{taches}} + \underbrace{\sum_{i=1}^{n} \pi \sum_{m=1}^{M+1} \mathbb{1}_{i\in\bigcup_{l=m+1}^{M+1} B_l}}_{f_{maint}}$$

où $\mathbb{1}$ désigne la fonction indicatrice.

On étudie ensuite séparément le terme relatif au tâches f_{taches} et le terme de maintenance f_{maint}. Dans les deux cas, on décompose la position i d'une tâche selon la position m du paquet dans lequel elle se trouve et position j de la tâche au sein du paquet : $i = j + \sum_{l=1}^{m-1} |B_l|$

$$
\begin{aligned}
f_{taches} &= \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \left(n - \left(j + \sum_{l=1}^{m-1} |B_l| \right) + 1 \right) p_{jm} \\
&= \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \left(\sum_{l=1}^{M+1} |B_l| - \sum_{l=1}^{m-1} |B_l| - j + 1 \right) p_{jm} \\
&= \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \left(\sum_{l=m}^{M+1} |B_l| - j + 1 \right) p_{jm} \\
&= \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \left(\sum_{l=m+1}^{M+1} |B_l| + |B_m| - j + 1 \right) p_{jm} \\
&= \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \sum_{l=m+1}^{M+1} |B_l| p_{jm} + \sum_{m=1}^{M+1} \underbrace{\sum_{j=1}^{|B_m|} (|B_m| - j + 1)\, p_{jm}}_{\gamma_m}
\end{aligned}
$$

En permutant les sommes, et en regroupant les durées des tâches dans le terme gauche, on obtient finalement :

$$
f_{taches} = \sum_{m=1}^{M+1} \gamma_m + \sum_{m=1}^{M+1} \left(\sum_{l=m+1}^{M+1} |B_l| \right) p(B_m)
$$

On s'intéresse ensuite au terme de maintenance :

$$
f_{maint} = \sum_{m=1}^{M+1} \sum_{j=1}^{|B_m|} \pi \sum_{k=1}^{M+1} \mathbb{1}_{j+\sum_{l=1}^{k-1} |B_l| \in \bigcup_{l=k+1}^{M} B_l}
$$

On constate qu'on peut simplifier l'indicatrice, qui vaut 1 si $m > k$ et 0 sinon. Alors :

$$
f_{maint} = \sum_{k=1}^{M+1} \sum_{m=k+1}^{M+1} \sum_{j=1}^{|B_m|} \pi = \pi \sum_{m=1}^{M+1} \sum_{l=m+1}^{M+1} |B_l|
$$

En additionnant les deux f_{taches} et f_{maint}, on obtient le résultat du lemme.

□

Ce résultat permet de démontrer une des propriétés structurelles présentées ci-après et est à la base du programme dynamique polynomial pour le cas du nombre de familles de tâches fixé présenté dans la sous-section 4.6.3

4.6.2 Structure des solutions optimales

On démontre dans cette section deux propriétés concernant la structure des solutions optimales. La première propriété consiste à observer qu'au sein d'un même paquet, trouver l'ordre optimal des tâches revient à résoudre $1||\sum C_i$, et donc que les tâches au sein des

paquets peuvent être triées sans perte de généralité avec l'ordre SPT. La seconde propriété concerne l'ordre des paquets au sein d'une solution optimale.

Le problème que nous étudions est une généralisation de $1, M(prev, h, per)||\sum C_i$, étudié par Qi et al. (1999). Leurs auteurs démontrent trois propriétés structurelles pour ce problème.

Proposition 4.6.2. *(Qi et al., 1999)*

Toute instance réalisable de $1, M(prev, h, per)||\sum C_i$ *admet une solution optimale telle que :*

1. *Les tâches au sein de chaque paquet sont triées selon la règle SPT*
2. *Les paquets sont triés selon l'ordre croissant de* $\frac{\pi+p(B_m)}{|B_m|}$
3. *Les paquets sont triés selon l'ordre croissant de* $\frac{p(B_m)}{|B_m|}$

Nous montrons dans cette section que les deux premières propriétés sont conservées pour $1, M(pred, h, ehi, perf)||\sum C_i$, mais que la troisième n'est plus vérifiée.

Proposition 4.6.3. *Toute instance réalisable de* $1, M(pred, h, ehi, perf)||\sum C_i$ *admet une solution optimale telle que les tâches au sein de chaque paquet sont triées selon la règle SPT.*

Démonstration. Supposons qu'il existe une solution optimale S telle que

$$\exists m, \exists j, k \in B_m, s_j < s_k \text{ et } p_j > p_k$$

Construisons S', solution telle que j, k soient interverties.

$$\sum_i C'_i = \sum_i C_i + \underbrace{(p_k - p_j)}_{<0} < \sum_i C_i$$

Ce qui contredit l'optimalité de S.

□

On suppose les paquets de tâches connus et on connaît l'ordonnancement des tâches au sein de ces paquets. Le résultat suivant est une règle optimale d'ordonnancement des paquets.

Proposition 4.6.4. *Toute instance réalisable de* $1, M(pred, h, ehi, perf)||\sum C_i$ *admet une solution optimale telle que les paquets sont triés selon l'ordre croissant de* $\frac{\pi+p(B_m)}{|B_m|}$

Démonstration. Soient deux paquets B_q et B_{q+1} planifiés successivement dans une solution de délai moyen $\sum C_i$. Supposons que la solution telle que $B'_q = B_{q+1}$, $B'_{q+1} = B_q$ et $\forall m \notin \{q, q+1\}$, $B'_m = B_m$ admette un délai moyen $\sum C'_i$.

On applique le lemme 4.6.1 sur $\sum C'_i$ et $\sum C_i$, afin de calculer la différence entre les deux termes. Les termes $\sum_{m=1}^{M+1} \gamma_m$ s'annulent. On décompose alors cette différence de la manière suivante. On extrait spécifiquement les termes de la sommes correspondant à $m \in q, q+1$ dans un terme $f_{q,q+1}$ que l'on calcule séparément du reste de la somme f_{autres}.

$$\sum C'_i - \sum C_i = f_{q,q+1} + f_{autres}$$

où

$$
\begin{aligned}
f_{q,q+1} &= (p(B_q)' + \pi)\left(\sum_{l=q+1}^{M+1} |B'_l|\right) + (p(B_{q+1})' + \pi)\left(\sum_{l=q+2}^{M+1} |B'_l|\right) \\
&\quad - (p(B_q) + \pi)\left(\sum_{l=q+1}^{M+1} |B_l|\right) - (p(B_{q+1}) + \pi)\left(\sum_{l=q+2}^{M+1} |B_l|\right) \\
&= (p(B_{q+1}) + \pi)\left(|B_q| + \sum_{l=q+2}^{M+1} |B_l|\right) + (p(B_q) + \pi)\left(\sum_{l=q+2}^{M+1} |B_l|\right) \\
&\quad - (p(B_q) + \pi)\left(|B_{q+1}| + \sum_{l=q+2}^{M+1} |B_l|\right) - (p(B_{q+1}) + \pi)\left(\sum_{l=q+2}^{M+1} |B_l|\right) \\
&= |B_q|\,(p(B_{q+1}) + \pi) - |B_{q+1}|\,(p(B_q) + \pi) \\
f_{autres} &= \sum_{m \notin \{q,q+1\}} (p(B_m) + \pi)\left(\sum_{l=1}^{m+1} |B'_l| - \sum_{l=1}^{m+1} |B_l|\right)
\end{aligned}
$$

Pour f_{autres}, les termes pour $m < q$ s'annulent et les termes pour $l \notin \{q, q+1\}$ s'annulent également. Il reste :

$$f_{autres} = \sum_{m=q+2}^{M+1} (p(B_m) + \pi)\left(|B'_q| + |B'_{q+1}| - |B_q| - |B_{q+1}|\right) = 0$$

D'où :

$$\sum C'_i - \sum C_i = |B_q|\,(p(B_{q+1}) + \pi) - |B_{q+1}|\,(p(B_q) + \pi)$$

On a alors

$$\sum C_i \leq \sum C'_i \iff \frac{\pi + p(B_q)}{|B_q|} \leq \frac{\pi + p(B_{q+1})}{|B_{q+1}|}$$

Ce qui signifie que la solution ayant la plus petite valeur est celle telle que les paquets B_q et B_{q+1} soient triés selon l'ordre croissant de $\frac{\pi + p(B_m)}{|B_m|}$, d'où la règle.

□

Nous avons montré ici que les deux premiers résultats de la proposition 4.6.2 concernant le problème d'ordonnancement avec indisponibilités $1, M(prev, h, per)||\sum C_i$ restent vérifiés pour notre problème $1, M(pred, h, ehi, perf)||\sum C_i$.

La dernière propriété n'est au contraire plus vérifiée, comme en atteste le contre-exemple suivant. On considère $n = 3$ tâches, et une santé maximale $H_{max} = 100$: la tâche J_1 a une durée $p_1 = 1$ et une usure $w_1 = 100$, tandis que les tâches J_2 et J_3 ont la même durée $p_2 = p_3 = 2$ et la même usure $w_2 = w_3 = 50$. De plus nous supposons que la durée de la maintenance est $\pi = 10$. Les deux solutions optimales pour cette instance consistent à construire deux paquets $B_1 = \{J_2, J_3\}$ et $B_2 = \{J_1\}$ dans cet ordre, séparés par une maintenance. Dans ce cas on a $\frac{p(B_1)}{|B_1|} = 2 > \frac{p(B_2)}{|B_2|} = 1$, ce qui contredit la règle.

4.6.3 Cas des familles de tâches [1]

Dans les sections précédentes, aucune hypothèse sur la structure des tâches n'a été faite : la santé est décorrelée du temps et les tâches sont quelconques et complètement indépendantes. Ici, nous nous intéressons à un cas particulier de la minimisation de $\sum C_i$, où il est possible de regrouper les tâches, en un nombre limité de familles de tâches d'usure identiques. En pratique, dans le milieu industriel, un équipement va être souvent amené a effectuer un ensemble limité de tâches identiques un grand nombre de fois, ce qui est cohérent avec l'hypothèse proposée dans cette section.

Formellement, les tâches sont regroupées par classe d'équivalence d'usure Ψ_f, appelées familles de tâche. On note $J_{f,i}$ la i-ième tâche de la f-ième famille. $\Psi_f = \{J_{f,1} \dots J_{f,n_f}\}$. On note F le nombre de familles de tâches d'une instance, n_f le nombre de tâches appartenant à la famille Ψ_f et p_{fi} la durée de la i-ième tâche de la famille. On note ce problème $\mathcal{P}_F = 1, M(pred, h, ehi, perf)|w_f|\sum C_i$.

Lemme 4.6.5. *Toute instance réalisable de* $1, M(pred, h, ehi, perf)|w_f|\sum C_i$ *admet une solution optimale telle que les tâches de chaque famille soient triées selon l'ordre SPT.*

Démonstration. Supposons qu'il existe une solution optimale S telle que

$$\exists f \in F, \exists j, k \in \Psi_f, s_j < s_k \text{ et } p_j > p_k$$

Construisons S', solution telle que j, k soient interverties. Le planning reste réalisable puisque les deux tâches ont la même usure.

$$\sum_i C'_i = \sum_i C_i + \underbrace{(p_k - p_j)}_{<0} < \sum_i C_i$$

Ce qui contredit l'optimalité de S.

□

Nous proposons ci-après deux formulations de programmation dynamique, une pseudo-polynomiale et une polynomiale permettant de modéliser ce problème. Pour chacune d'entre-elles, nous commençons par présenter le cas à deux familles, puis proposons leur extension au cas à F familles.

Formulation pseudo-polynomiale La première formulation, pseudo-polynomiale se base sur le résultat suivant, introduit dans la sous-section 4.6.1 pour la démonstration du Lemme 4.6.1.

$$\sum_{i\in\mathcal{J}} C_i = \underbrace{\sum_{i=1}^{n}(n-i+1)p_i}_{f_{taches}} + \underbrace{\sum_{i=1}^{n}\pi\sum_{m=1}^{M}\mathbb{1}_{i\in\bigcup_{l=m+1}^{M}B_l}}_{f_{maint}} \tag{4.6.1}$$

1. Une partie de ces résultats ont été obtenus en collaboration avec Vincent Fagnon

Ainsi, ajouter une tâche au début de l'ordonnancement augmente de sa durée la valeur de $\sum C_i$ et augmente de sa durée la valeur de C_i pour chacune des futures tâches. Ajouter une tâche de maintenance décale de même la valeur de C_i pour toutes les tâches suivantes.

Le schéma de programmation dynamique se résume ainsi : chaque décision consiste à rajouter une tâche de production ou de maintenance au début de l'ordonnancement. Cela augmente la valeur de $\sum C_i$ par sa durée autant de fois qu'il reste de tâches à planifier et réduit la valeur de la santé dans le cas de la production ; la réinitialise dans le cas de la maintenance. On peut ensuite répéter avec la tâche suivante, jusqu'à ce que toutes les tâches aient été planifiées.

On suppose que les tâches sont pré-triées selon la règle SPT. Avec le Lemme 4.6.5, on voit qu'on peut limiter le choix entre les tâches les plus courtes de chaque famille, et ainsi ne mémoriser que le nombre de tâches restantes à planifier.

Un état est ainsi décrit par le nombre de tâches restantes pour chaque famille et un niveau de santé. Dans le cas $F = 2$, la fonction $f_d^*(\nu_1, \nu_2, H)$ décrit la valeur optimale de $\sum C_i$ pour les ν_1 dernières tâches de la famille Ψ_1, ν_2 dernières tâches de la famille Ψ_2 et un niveau de santé H. On cherche à résoudre $f_d^*(n_1, n_2, H)$ et les cas de base sont $f_d^*(0, 0, H) = 0$ pour toute valeur de H.

La formule de récurrence est alors la suivante :

$$f_d^*(\nu_1, \nu_2, H) = min \begin{cases} p_{1,n_1-\nu_1}(\nu_1 + \nu_2) + f_d^*(\nu_1 - 1, \nu_2, H - w_1) & \text{si } H - w_1 > 0 \\ p_{2,n_2-\nu_2}(\nu_1 + \nu_2) + f_d^*(\nu_1, \nu_2 - 1, H - w_1) & \text{si } H - w_2 > 0 \\ \pi(\nu_1 + \nu_2) + f_d^*(\nu_1, \nu_2, H_{max}) & \text{si } H < H_{max} \end{cases}$$

Cette formule peut être étendue à un nombre quelconque de famille F. Pour ce faire, on introduit les notations vectorielles suivantes $\overrightarrow{n} = [n_1 \dots n_F]^\top$ et $\overrightarrow{\nu} = [\nu_1 \dots \nu_F]^\top$, et on note $\overrightarrow{e_f}$ le f-ième vecteur de base dans $\mathbb{R}^F$. On note également $||\cdot||_1$ la norme 1 dans cet espace. Les cas de base sont la forme $f_d^*(\vec{0}, H) = 0$ et on cherche à calculer $f_d^*(\overrightarrow{n}, H_{max})$. La formule de récurrence pour le cas général devient :

$$f_d^*(\overrightarrow{\nu}, H) = min \begin{cases} min_{f \in F} \left\{ p_{f,n_f-\nu_f} ||\overrightarrow{\nu}||_1 + f_d^*(\overrightarrow{\nu} - \overrightarrow{e_f}, H - w_f) | H - w_f > 0 \right\} \\ \pi ||\overrightarrow{\nu}||_1 + f_d^*(\overrightarrow{\nu}, H_{max}) \quad \text{si } H < H_{max} \end{cases}$$

Le tableau de programmation dynamique admet $H_{max} \prod_{f=1}^{F} n_f$ cases, chacune reliée par la formule de récurrence à au plus $(F + 1)$ prédécesseurs. Par application de l'inégalité arithmético-géométrique :

$$\prod_{f=1}^{F} n_f \leq \left(\frac{1}{F} \sum_{f=1}^{F} n_f \right)^F = \left(\frac{n}{F} \right)^F$$

La complexité de la résolution à l'aide de ce programme dynamique est donc pseudo-polynomiale $O(\frac{F+1}{F^F} n^F H_{max})$.

Formulation polynomiale Nous proposons une deuxième formulation de programmation dynamique, cette fois-ci basée sur la décomposition du planning par paquets de tâches du Lemme 4.6.1. La décision à chaque étape consiste à définir de combien de tâches de chaque famille est constituée le prochain paquet. On sait alors d'après le lemme que la valeur optimale du délai moyen se décompose en trois termes : le reste du planning est égal au délai moyen des tâches du paquet trié selon la règle SPT, l'impact du paquet sur le délai des tâches restantes et la valeur optimale du délai moyen pour les tâches restantes. Pour le cas $F = 2$, on note $f_b^*(\nu_1, \nu_2)$ l'optimum du délai moyen pour les ν_1 dernières tâches de la famille Ψ_1 et les ν_2 dernières tâches de la famille Ψ_2. L'unique cas de base est $f_b^*(0,0) = 0$, et on cherche à calculer $f_b^*(n_1, n_2)$. La formule de récurrence s'écrit de la manière suivante :

$$\begin{aligned} f_b^*(\nu_1,\nu_2) = min_{(i,j)\in\Theta(\nu_1,\nu_2)} \Bigg\{ & \Gamma\left(\left(\bigcup_{k=n_1-\nu_1}^{n_1-\nu_1+i}\{J_{1,k}\}\right)\cup\left(\bigcup_{k=n_2-\nu_2}^{n_2-\nu_2+j}\{J_{2,k}\}\right)\right) \\ & + (\nu_1+\nu_2-i-j)\left[\pi + \sum_{k=n_1-\nu_1}^{n_1-\nu_1+i} p_{1,k} + \sum_{k=n_2-\nu_2}^{n_2-\nu_2+j} p_{2k}\right] \\ & + f_b^*(\nu_1-i,\nu_2-j)\Bigg\} \end{aligned}$$

$$\text{où} \qquad \Theta(\nu_1,\nu_2) = \{(i,j) \in (\{0\dots\nu_1\}\times\{0\dots\nu_2\}) \setminus (0,0) | iw_1 + jw_2 \leq H_{max}\}$$

et $\Gamma(\mathcal{J})$ représente la valeur de l'optimum de $1||\sum C_i$ pour les tâches de l'ensemble $\mathcal{J}$, c'est-à-dire, la valeur du délai moyen lorsque les tâches sont triées selon la règle SPT.

De la même manière que pour la formulation pseudo-polynomiale, ce programme dynamique peut être étendu à un nombre quelconque de familles à l'aide des notations vectorielles introduites précédemment. Le cas de base devient $f_b^*(\vec{0}) = 0$ et on cherche à calculer $f_b^*(\overrightarrow{n})$. La formule de récurrence s'écrit de la manière suivante :

$$\begin{aligned} f_b^*(\overrightarrow{\nu}) = min_{\vec{i}\in\Theta(\overrightarrow{\nu})} \Bigg\{ & \Gamma\left(\bigcup_{f=1}^{F}\left(\bigcup_{k=n_f-\nu_f}^{n_j-\nu_f+i_f}\{J_{f,k}\}\right)\right) \\ & + ||\overrightarrow{\nu}-\vec{i}||_1\left[\pi + \sum_{f=1}^{F}\sum_{k=n_f-\nu_f}^{n_f-\nu_f+i_f} p_{f,k}\right] \\ & + f_b^*(\overrightarrow{\nu}-\vec{i})\Bigg\} \end{aligned}$$

$$\text{où} \qquad \Theta(\overrightarrow{\nu}) = \left\{\vec{i} \in \prod_{f=1}^{F}\{0\dots\nu_f\}\setminus\vec{0} \;\middle|\; \sum_{f=1}^{F} i_f w_f \leq H_{max}\right\}$$

On utilise l'inégalité arithmético-géométrique pour borner le nombre d'états $O\left(\left(\frac{n}{F}\right)^F\right)$ et le nombre de transitions par $O\left(\left(\frac{n}{F}\right)^F\right)$, la complexité totale de l'algorithme de programmation dynamique s'écrit $O\left(\left(\frac{n}{F}\right)^{2F}\right)$.

Le problème $\mathcal{P}_F = 1, M(pred, h, ehi, perf)|w_f| \sum C_i$ est donc polynomial pour tout nombre fixé de familles.

4.7 Conclusion

Dans ce chapitre, nous introduisons un nouveau problème d'ordonnancement conjoint de la production et de la maintenance conditionnée par la santé des composants de l'équipement. Nous commençons par proposer un système de notations généralisées permettant de classer la littérature de l'ordonnancement conjoint de la production et de la maintenance, et explicitons le positionnement de ce nouveau problème vis-à-vis de la littérature.

Nous proposons dans un second temps une cartographie de la complexité et des différentes propriétés des variantes du problème. Nous expliquons ensuite comment les modéliser en PLNE.

Enfin, nous approfondissons le cas de la minimisation du délai maximum C_{max} et du délai moyen $\sum C_i$. Dans le cas de $\sum C_i$, nous étudions la structure des solutions optimales et dérivons des programmes dynamiques pour le cas où les tâches peuvent être regroupées en familles. Dans le cas de C_{max}, nous établissons des liens avec les problèmes de Bin-Packing et dérivons des résultats d'approximabilité. Ces liens entre notre problème d'ordonnancement et le Bin-Packing sont approfondis dans le chapitre 5 de ce manuscrit.

Chapitre 5

Approche bin-packing du problème d'ordonnancement

Sommaire

5.1 Introduction

Dans le chapitre précédent, nous avons étudié un problème d'ordonnancement conjoint de la production et de la maintenance basé sur un indicateur de santé des équipements. Nous avons établi que le problème de minimisation du délai maximum, sous certaines hypothèses, revient à regrouper les tâches entre les périodes de maintenance de la manière la plus efficace possible, afin que le nombre de maintenances soit minimisé. Ce sous-problème est donc une variante du problème de Bin-Packing (BPP). Nous nous intéressons à présent plus spécifiquement au cas où chaque tâche possède un seuil de santé minimum devant être respecté au moment où elle est exécutée par la machine.

Dans ce chapitre, ce sous-problème, appelé Bin-Packing avec Seuils (BPS), est étudié de manière plus approfondie. Dans un premier temps, le problème est formalisé, positionné dans la littérature du BPP et plusieurs résultats théoriques sont présentés. Ensuite, trois approches de la littérature basées sur la programmation linéaire en nombres entiers sont étendues à ce problème et leurs performances sont comparées au moyen d'expérimentations sur un benchmark.

Le problème et les résultats présentés dans ce chapitre font l'objet d'une publication soumise en revue. Une partie des résultats présentés dans ce chapitre ont été obtenus lors du stage de master de Gérémi Bridonneau, que j'ai co-encadré (Bridonneau, 2022).

Le chapitre est organisé comme suit : dans la section 5.2, le problème est formalisé et une visualisation est proposée sur un exemple. Ensuite, dans la section 5.3, le positionnement du problème et de la méthodologie proposée vis-à-vis de la littérature du Bin-Packing est clarifié. Quelques propriétés sur la forme des solutions du problème et la performance des algorithmes online sont présentées et prouvées dans la section 5.4. Une première formulation de programmation linéaire en nombres entiers faisant intervenir un nombre polynomial de variables est présenté dans la section 5.5. Dans la section 5.6, une méthode de génération de colonnes basée sur une formulation exponentielle est présentée. Une formulation pseudo-polynomiale de type arc-flot est étudiée dans la section 5.7. Enfin, les différentes méthodes ont été testées et comparées sur un jeu d'instances adapté de la littérature. Les résultats sont présentés dans la section 5.8.

5.2 Définition du problème

Le problème du Bin-Packing avec Seuils est une extension du Bin-Packing classique. Formellement, le problème consiste à partitionner un ensemble de n items $i \in \mathcal{N} = \{J_1 \dots J_n\}$ en un ensemble de cardinalité minimale de boîtes $\mathcal{B} = \{B_1 \dots B_u\}$. Une solution du problème est entièrement décrite comme la partition $\mathcal{B}$ de l'ensemble $\mathcal{N}$ avec un ordre total σ_b pour chacun des sous-ensembles B_b de la solution. Chaque item J_i possède un poids $w_i \in \mathbb{N}$ et un seuil $t_i \in \mathbb{N}$. Les boîtes sont de capacités toutes identiques c. Le remplissage des boîtes doit respecter les contraintes de capacité et de seuils définies ci-après.

Exemple illustratif Nous donnons une première intuition du sens de ces contraintes sur un exemple, puis nous précisons leurs définitions dans un second temps.

Nous considérons l'instance suivante :

Item	J_1	J_2	J_3	J_4
w_i	2	1	1	2
t_i	3	3	2	3

La Figure 5.1 et la Figure 5.2 représentent respectivement une solution réalisable et une solution non-réalisable pour cette instance. Sur cette visualisation, chaque item est représenté par un rectangle décomposé en deux parties : la partie plus foncée correspond à l'item en lui-même et la hauteur de ce rectangle est égale au poids de l'item, la partie de la boîte correspondante est donc occupée par l'item et ne peut pas contenir un autre item en même temps. La hauteur totale du rectangle (partie foncée et partie claire) correspond au seuil de l'item. La contrainte de seuil impose que la capacité restante dans la boîte au moment où l'item est placé doit être supérieure à son seuil. Si la partie claire dépasse par le bas de la boîte, alors cela signifie que la contrainte de seuil n'est pas respectée.

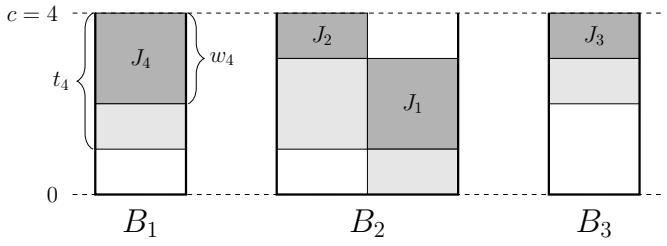


FIGURE 5.1 – Solution réalisable pour l'exemple

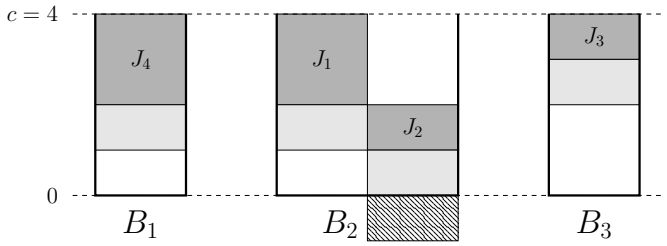


FIGURE 5.2 – Solution non-réalisable pour l'exemple

Pour la solution représentée sur la Figure 5.1, le poids total dans chaque boîte est inférieur à la capacité, la contrainte de capacité est donc bien vérifiée. Quant à la contrainte de seuil, on vérifie bien $t_4 = 3 \leq c = 4$ pour B_1, $t_2 = 3 \leq c = 4$ et $t_1 = 3 \leq c - w_2 = 3$ pour B_2, et $t_3 = 2 \leq c = 4$ pour B_4. La solution est donc bien réalisable.

En revanche, dans la solution représentée sur la Figure 5.2, les items J_1 et J_2 ont été inversés. La contrainte de capacité n'est pas impactée. En revanche, on observe une violation de la contrainte de seuil par J_2, car $t_2 = 3 > c - w_1 = 2$, la solution n'est donc pas réalisable. La violation correspond à la partie hachurée sur la figure.

Les deux solutions proposées contiennent trois boîtes, il est possible de trouver une meilleure partition ne contenant que deux boîtes en regroupant les items J_3 et J_4 dans une même boîte. La solution correspondante est représentée sur la Figure 5.3.

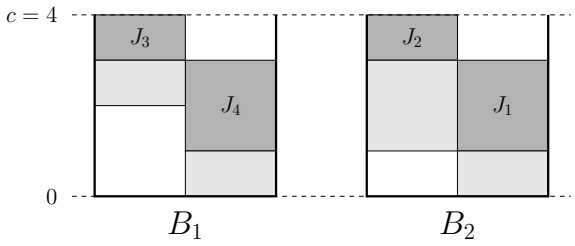


FIGURE 5.3 – Solution optimale pour l'exemple

Par rapport au problème d'ordonnancement initial, les poids des items représentent l'usure causée par les tâches sur l'équipement et les seuils une quantité minimale de santé nécessaire pour que la tâche puisse être exécutée sans risque.

Définition des contraintes Nous présentons à présent les définitions formelles des deux contraintes du problème.

La contrainte de capacité impose que la somme des poids au sein de chaque boîte ne dépasse pas la capacité $c \in \mathbb{N}$, identique pour toutes les boîtes. Formellement, cette contrainte peut s'écrire de la manière suivante :

$$\forall B_b \in \mathcal{B},\ \sum_{i \in B_b} w_i \leq c \tag{5.1}$$

La contrainte de seuil, quant à elle, impose que pour chaque remplissage partiel d'une boîte B_b, la capacité restante soit supérieure au seuil du prochain item de la boîte.

$$\forall B_b \in \mathcal{B},\ \forall i \in B_b,\ c - \sum_{j=1}^{i-1} w_{\sigma_b^{-1}(j)} \geq t_i \tag{5.2}$$

En tant que généralisation du Bin-Packing classique, ce problème est NP-complet au sens fort.

5.3 Positionnement

Le problème du Bin-Packing (BPP) et son équivalent Cutting Stock (CSP) ont fait l'objet de très nombreux travaux de recherche. Dans cette section, nous proposons une revue non-exhaustive des variantes présentant des liens avec le problème étudié ici et positionnons les méthodes de résolution proposées par rapport aux méthodes de la littérature.

5.3.1 Variantes du bin-packing

Dans sa version la plus simple, le BPP consiste à partitionner un ensemble d'items de poids différents en un ensemble de boîtes de capacité fixée, de telle sorte que le poids total des items dans chaque boîte n'excède pas la capacité. Le CSP est équivalent, mais formulé de manière différente. Un ensemble de pièces de taille différentes doit être découpé dans des plaques de longueurs fixées, de sorte à minimiser l'ensemble de plaques utilisées. Dans les instances de CSP, les pièces (ou items) sont décrits avec leur demande (ou multiplicité), tandis que pour les instances de BPP, deux items identiques sont décrits séparément.

Un ensemble très large de variantes online et offline de ces problèmes a été étudié au cours des dernières décennies. Coffman et al. (2013) proposent une revue et une méthode de classification de ces variantes.

La principale originalité du BPS par rapport au BPP est la notion d'ordre des items dans les boîtes qui a un impact sur la faisabilité de la solution.

Du côté des variantes online, on peut citer le Ordered Open-End Bin-Packing Problem, où les items sont libérés les uns à la suite des autres, et où le dernier item d'une boîte peut excéder la capacité (Balogh et al., 2021). Des ordres partiels sur les boîtes ont également été considérés pour des problèmes d'équilibrage de chaîne d'assemblage (Wee & Magazine, 1982) et pour des problèmes d'ordonnancement avec contraintes de ressource (Garey et al., 1976). Cependant, à notre connaissance, il n'existe pas de travaux dans la littérature autour de variantes offline de BPP ou CSP pour lesquelles l'ordre des items dans les boîtes a une importance.

5.3.2 Méthodes de résolution

Les méthodes exactes et modèles de programmation mathématique ont fait l'objet d'une revue de littérature très large par Delorme et al. (2016). Concernant les méthodes approchées, une revue des algorithmes d'approximation pour le BPP et ses variantes online et offline a été proposée par Coffman et al. (2013).

La plupart des méthodes exactes de la littérature reposent sur la programmation mathématique. Du côté de la programmation linéaire, de nombreux modèles faisant intervenir un nombre polynomial, pseudo-polynomial ou exponentiel de variables ont été considérés.

Le modèle polynomial est traditionnellement attribué à Kantorovich (1960), et sert habituellement de définition au problème. Cependant, en raison d'une relaxation lagrangienne assez mauvaise, il est peu utilisé en pratique pour la résolution.

Une très grande variété de modèles pseudo-polynomiaux, faisant intervenir un nombre de variables dépendant de la capacité des boîtes, ont été étudiés. Le premier modèle de ce type est connu sous le nom de One-Cut, et a été découvert indépendamment par Rao (1976) et Dyckhoff (1981). L'idée principale est que lorsqu'un item est placé, la boîte peut être divisée en deux parties, la partie contenant l'item et un résidu de plus petite taille dans lequel d'autres items pourront éventuellement être placés. Les auteurs introduisent alors des variables x_{pq} comptant le nombre de résidus de taille q découpés dans un résidu de taille p. L'objectif est alors de minimiser le nombre de résidus découpés dans une boîte vide, ce qui est égal au nombre de boîtes total.

Cambazard et O'Sullivan (2010) ont proposé une formulation appelée DP-flow pour le problème de bin-packing, où les variables correspondent aux décisions dans le programme dynamique.

Les autres formulations pseudo-polynomiales existantes appartiennent à la famille des formulations arc-flot. Elles consistent à formuler le problème comme un problème de flot dans un graphe orienté pour dériver un modèle de programmation linéaire en nombres entiers. Contrairement à la formulation One-Cut, permettant de résoudre spécifiquement le BPP et le CSP, les formulations arc-flot ont été utilisées pour modéliser et résoudre une très large variété de problèmes, notamment en ordonnancement et pour les problèmes de tournées de véhicules. Une revue sur les fondements théoriques et les applications des modèles arc-flot a été proposée par De Lima et al. (2022). En plus d'être assez faciles d'implémentation, ces méthodes sont également parmi les plus efficaces de la littérature pour la résolution exacte du BPP et du CSP.

La première formulation de ce type, proposée par Valério de Carvalho (2002), présente de forts liens avec le schéma de programmation dynamique du problème de sac à dos. Le remplissage d'une boîte est représenté par un chemin dans un graphe orienté dont les sommets représentent la capacité restante de la boîte. La solution de bin-packing est alors représentée comme un flot au sein de ce graphe. Les variables correspondent au flot à travers les arcs de ce graphe et l'objectif à minimiser est la valeur de ce flot. Le modèle présente un très grand nombre de symétries, plusieurs améliorations ont été proposées pour réduire le nombre de sommets.

Plus récemment Brandão et Pedroso (2016) ont proposé une formulation différente avec une construction de graphe plus compacte. En pratique, cette méthode, plus sophistiquée, figure parmi les méthodes les plus compétitives pour la résolution du BPP et du CSP.

Enfin, Delorme et Iori (2020) ont proposé une amélioration du modèle de Valério de Carvalho qui consiste à replier le graphe sur lui-même pour diviser par deux le nombre de sommets. On retrouve alors deux types d'arcs : les arcs directs, non affectés par la transformation et des arcs réfléchis, qui reviennent en arrière à cause du pliage. Une boîte réalisable est alors décrite par une paire de chemins, un contenant un arc réfléchi, un autre n'en contenant pas, se rejoignant sur le même sommet.

Pour notre problème, nous avons opté pour une adaptation du modèle de Valério de Carvalho, notamment en raison de sa simplicité et de son efficacité. Les autres modèles ne sont pas directement adaptables puisqu'ils exploitent des brisures de symétrie incompatibles pour la contrainte de seuil. Le modèle de Brandão et Pedroso utilise un pré-tri des items par ordre de poids, et la fusion d'une partie des sommets dans le modèle de Delorme et Iori entraine des complications pour la modélisation de la contrainte de seuil.

Les approches basées sur des modèles exponentiels sont également très efficaces pour la résolution du BPP et du CSP. Le modèle set-covering et son équivalent set-partitioning ont été introduits par Gilmore et Gomory (1961). Ils reposent sur l'énumération de l'ensemble des combinaisons réalisables d'items au sein d'une même boîte, appelées colonnes. Une solution réalisable au BPP/CSP est alors un ensemble de colonnes qui couvre (ou partitionne) l'ensemble des items de l'instance. En raison du nombre exponentiel de variables, ces modèles ne sont pas utilisés pour la résolution directe. Néanmoins, ils sont connus pour avoir une excellente relaxation linéaire qui peut être calculée efficacement à l'aide d'une procédure de

génération de colonnes (Gilmore & Gomory, 1961). Cette méthode consiste à résoudre le modèle à chaque itération en considérant un ensemble de colonnes restreint puis de chercher une nouvelle colonne améliorante jusqu'à arriver à l'optimum fractionnaire. Une méthode de génération de colonnes alternative, basée sur une formulation arc-flot, a été proposée par Valério de Carvalho (1999).

La génération de colonnes ne fournit qu'une borne inférieure et ne produit pas de solution réalisable entière. Néanmoins, elle est la base de nombreuses heuristiques et méthodes exactes. La méthode exacte branch-and-price combine des techniques de branchement avec la génération de colonnes. Une fois la solution fractionnaire obtenue, des branchements sur les variables non-entières sont combinés avec des itérations supplémentaires de génération de colonnes pour parvenir à reconstruire une solution entière optimale (Vance, 1998). Il existe une littérature très riche sur les algorithmes branch-and-price pour le BPP et CSP, une revue non-exhaustive de ces travaux est proposée dans la revue de littérature Delorme et al. (2016). L'algorithme branch-and-cut-and-price proposé par Belov et Scheithauer (2006) est, à l'heure actuelle, l'une des méthodes exactes les plus efficaces pour la résolution du BPP et du CSP.

Très récemment, De Lima et al. (2023) ont proposé un schéma de branch-and-price exploitant la décomposition de Dantzig-Wolfe du modèle arc-flot de Valério de Carvalho, des techniques de fixation de variables et une méthode de branchement spécifique, permettant d'obtenir de très bons résultats pour le CSP.

Un panorama des différentes méthodes heuristiques pour la reconstruction d'une solution à la fin de la procédure de génération de colonnes a été proposé par Sadykov et al. (2019). La méthode la plus utilisée est connue sous le nom d'heuristique du problème maître restreint, ou encore price-and-branch. Elle consiste à résoudre le programme linéaire en nombre entier avec l'ensemble des colonnes obtenues lors de la génération de colonnes. Son principal avantage est sa facilité d'implémentation une fois l'algorithme de génération de colonnes implémenté. Les heuristiques d'arrondis consistent à chercher une solution entière en arrondissant les variables non-entières obtenues à la fin de la génération de colonnes. Les heuristiques de diving consistent à explorer seulement une partie de l'arbre de décision du branch-and-price : après avoir obtenu une solution fractionnaire à l'aide de la génération de colonnes, une variable non-entière est sélectionnée et sa valeur est arrondie et fixée. La génération de colonnes est alors relancée et le processus continue jusqu'à obtenir une solution entière. Plusieurs travaux récents ont permis de montrer l'efficacité des heuristiques de diving sur des variantes du BPP et du CSP (Clautiaux et al., 2019 ; Sadykov & Vanderbeck, 2013). Dans le cadre de notre problème, nous proposons une adaptation de la génération de colonnes proposée par Gilmore et Gomory, combinée à l'heuristique de problème maître restreint pour la reconstruction d'une solution entière.

D'autres méthodes, non-basées sur la programmation linéaire ont également été développées. On peut citer en particulier BISON, un solveur basé sur branch-and-bound (Scholl et al., 1997) et les travaux de P. Shaw (2004) sur la contrainte globale Bin-Packing en programmation par contraintes. En pratique, ces méthodes sont moins efficaces que les méthodes basées sur la programmation linéaire en nombres entiers citées précédemment.

5.4 Propriétés fondamentales

Dans cette section, nous prouvons que l'on peut supposer sans perte de généralité que les seuils sont plus grands que les poids, et que les items sont triés par ordre décroissant $t_i - w_i$. Nous prouvons également que tout algorithme online a un ratio compétitif non borné.

Proposition 5.4.1. *Soit I une instance du BPS, et I' une version modifiée de I telle que $\forall i \in \mathcal{N},\ t'_i := max\{w_i, t_i\}$. S est une solution réalisable de I si et seulement si S est une solution réalisable I'.*

Démonstration. Soit $S = \{B_1 \dots B_u\}$ une solution réalisable de I. Les seuils n'apparaissant que dans les contraintes de seuils, il n'est pas nécessaire de vérifier la satisfaction d'autres contraintes que celles-ci. Par définition du maximum, $t_i \leq max\{w_i, t_i\}$, donc toute solution de I' est une solution de I ;

Soit $B_b(k)$ le k-ième item de la boîte B_b et $R_b(k) = c - \sum_{i=1}^{k-1} w_{B_b(i)}$ la capacité restante au moment d'ajouter le k-ième item de la boite. Les contrainte de seuil sont vérifiées si et seulement si :

$$\forall b \in \{1 \dots u\},\ \forall k \in \{1 \dots |B_b|\},\ R_b(k) \geq t_{B_b(k)} \tag{5.3}$$

De plus, la contrainte de capacité est respectée si et seulement si :

$$\begin{aligned}
&\forall b \in \{1 \dots u\}\ c - \sum_{i=1}^{|B_b|} w_{B_b(i)} \geq 0 \\
\iff &\forall b \in \{1 \dots u\},\ \forall k \in \{1 \dots |B_b|\},\ R_b(k) \geq \sum_{i=k}^{|B_b|} w_{B_b(i)} \\
\implies &\forall b \in \{1 \dots u\},\ \forall k \in \{1 \dots |B_b|\},\ R_b(k) \geq w_{B_b(k)}
\end{aligned}$$

Par conséquent, la contrainte de capacité et les contraintes de seuil impliquent que $R_b(k) \geq max\{t_{B_b(k)}, w_{B_b(k)}\}$ pour tout item J_k de toute boîte B_b. Ainsi, si S est une solution réalisable de I, alors S est également une solution de I'.

□

En conséquence de ce résultat, nous supposerons sans perte de généralité dans le reste de ce chapitre que les seuils sont supérieurs ou égaux au poids des items.

Proposition 5.4.2. *Toute instance du BPS admet une solution optimale telle que les items au sein de chaque boîte sont triés selon l'ordre décroissant de $t_i - w_i$.*

Démonstration. Ce résultat est prouvé au moyen d'un argument d'échange, illustré sur la Figure 5.4. Avec les notations de la preuve précédente, considérons une instance I du BPS et une solution optimale S telle qu'il existe deux items consécutifs J_i puis J_k au sein d'une même boîte B_b vérifiant $t_i - w_i < t_k - w_k$. Notons S' la solution où J_i et J_k sont inversés.

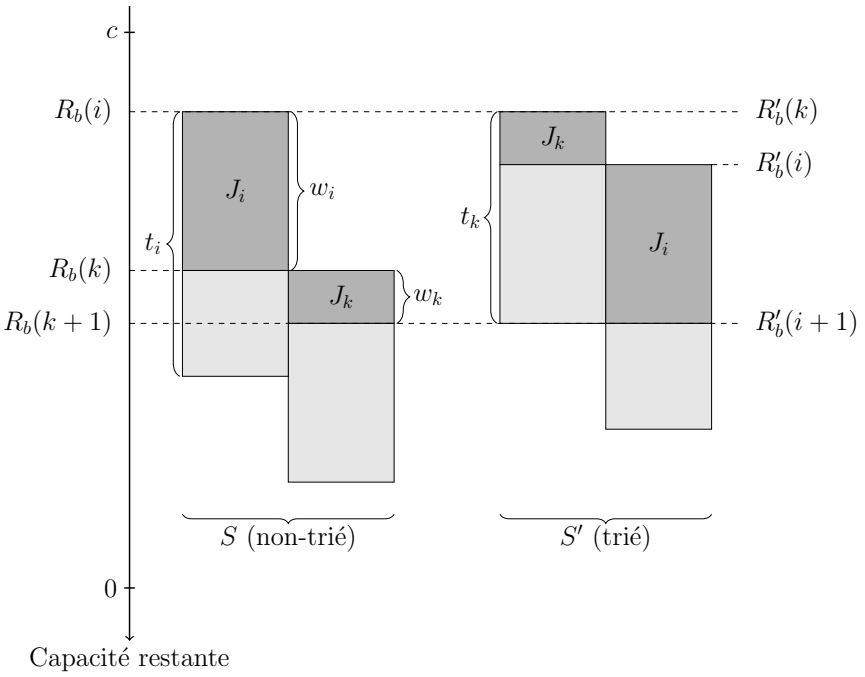


FIGURE 5.4 – Illustration de l'argument d'échange

Les capacités restantes avant et après les deux items sont inchangées par échange, il reste donc à vérifier la satisfaction des contraintes de seuil. Notons $R'_b(i)$ et $R'_b(k)$ les capacités restantes au moment de placer respectivement les items J_i et J_k dans la solution S'.

La faisabilité de S implique que $R_b(k) \geq t_k$. Par conséquent :

$$R'_b(k) = R_b(i) = R_b(k) + w_k \geq R_b(k) \geq t_k \tag{5.4}$$

La contrainte de seuil de l'item J_k est donc satisfaite. Prouvons également $R'_b(i) \geq t_i$. En utilisant l'hypothèse $t_i - w_i < t_k - w_k$ et la faisabilité de S, il est possible de dériver la suite d'inégalités suivantes :

$$t_i - w_i < t_k - w_k \tag{5.5}$$
$$\iff R_b(i) + t_i - w_i < R_b(i) + t_k - w_k \tag{5.6}$$
$$\iff R_b(k) + t_i < R_b(i) + t_k - w_k \tag{5.7}$$
$$\implies t_k + t_i < R_b(i) + t_k - w_k \tag{5.8}$$
$$\implies t_i < R_b(i) - w_k \tag{5.9}$$

Or $R_b(i) = R'_b(k)$, car ces deux quantités représentent la capacité restante avant de placer les deux items qui ont été échangés. En utilisant cette propriété dans l'inégalité précédente :

$$t_i < R'_b(k) - w_k = R'_b(i) \tag{5.9}$$

La contrainte de seuil est donc bien vérifiée pour l'item J_i dans S'. Les autres items de la boîte n'étant pas impactés par l'échange, leurs contraintes de seuil sont également vérifiées.

Donc S' est une solution réalisable pour l'instance I.

□

Dans le reste de ce chapitre, on suppose également sans perte de généralité que les items sont triés dans les boîtes selon l'ordre décroissant de $t_i - w_i$. Avec cette hypothèse, une solution est donc entièrement décrite par l'assignation des items aux boîtes.

Le dernier résultat de cette section concerne la performance des algorithmes online. Nous montrons que le nombre d'items est une borne inférieure du ratio compétitif de tout algorithme online, qui ne peut donc pas être borné par une constante.

Proposition 5.4.3. *Le ratio compétitif de tout algorithme online pour le BPS est non borné.*

Démonstration. La construction proposée pour montrer ce résultat s'appuie sur la suite de Fibonacci et ses propriétés. Nous rappelons donc dans un premier temps sa définition et un résultat utile pour la suite de la démonstration.

Définition 5.4.4. *La suite de Fibonacci* $(F_n)_{n\in\mathbb{N}}$ *est définie par* $F_0 = 0$, $F_1 = 1$ *et* $\forall n \in \mathbb{N},\ F_{n+2} = F_{n+1} + F_n$

Lemme 5.4.5. $\forall n \in \mathbb{N},\ \sum_{i=1}^{n} F_i = F_{n+2} - 1$

Démonstration. En utilisant la définition et l'annulation deux à deux des termes de la série par téléscopage :

$$\forall n \in \mathbb{N},\ \sum_{i=1}^{n} F_i = \sum_{i=1}^{n} (F_{i+2} - F_{i+1}) = F_{n+2} - F_2 = F_{n+2} - 1$$

□

Soit I_n l'instance du BPS à n items telle que pour tout item $J_i \in \mathcal{N}$, $w_i = F_{n-i+1}$ et $t_i = \sum_{k=1}^{i} w_k$, et telle que la capacité soit $c = \sum_{i=1}^{n} w_i$.

On peut facilement vérifier que pour toute valeur de n les instances I_n admettent une solution avec une seule boîte en triant les items par ordre décroissant d'index. La contrainte de capacité est serrée : $\sum_{i=1}^{k} w_i = c$. Les contraintes de seuil sont également toutes serrées : $\forall k \in \{J_1 \dots J_n\},\ c - \sum_{i=k+1}^{c} w_i = t_k$. La solution est donc bien réalisable. Cette solution pour I_5 est représentée sur la Figure 5.5.

Prouvons à présent par récurrence que tout algorithme online $\mathcal{A}$ renvoie une solution à n boîtes lorsque les items sont libérés un par un par ordre croissant d'index. Cette solution pour I_5 est représentée sur la Figure 5.6.

Notons, pour tout entier naturel k, P_k le prédicat "$\mathcal{A}$ utilise k boîtes pour ranger les k premiers items". Une boîte est nécessaire pour ranger le premier item, donc P_1 est immédiatement vérifié.

Prouvons $\forall k \in \{1 \dots n-1\},\ P_k \implies P_{k+1}$. En supposant P_k, montrons que l'item J_{k+1} ne rentre dans aucune des k premières boîtes, et donc qu'une nouvelle boîte doit être ouverte.

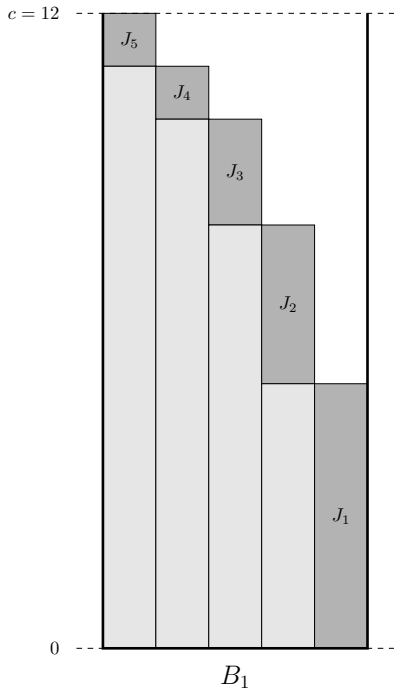


FIGURE 5.5 – Solution optimale offline pour l'exemple

$$\forall 1 < j \leq k,\ c - w_j = \sum_{i=1}^{n} w_i - w_j \tag{5.10}$$

$$= \sum_{i=1}^{k} w_i + \sum_{i=k+1}^{n} w_i - w_j \tag{5.11}$$

$$= t_k + \sum_{i=1}^{n-k} F_i - F_{n-j+1} \tag{5.12}$$

En utilisant le lemme 5.4.5 et la croissance stricte de la suite de Fibonacci :

$$\forall 1 < j \leq k,\ c - w_j = t_k + F_{n-k+2} - 1 - F_{n-j+1} \leq t_k - 1 < t_k \tag{5.13}$$

La capacité restante dans toutes les boîtes déjà ouvertes est donc inférieure au seuil de l'item J_k, il est donc nécessaire d'ouvrir une nouvelle boîte. Le prédicat P_{k+1} est donc vérifié.

Nous avons prouvé que pour toute valeur de n, l'instance I_n admet une solution optimale d'objectif 1, tandis que tout algorithme online fournit une solution d'objectif n lorsque les items sont libérés par ordre décroissant d'index. Le ratio compétitif de tout algorithme online admet donc n pour borne inférieure, et n'admet donc pas de borne supérieure constante.

□

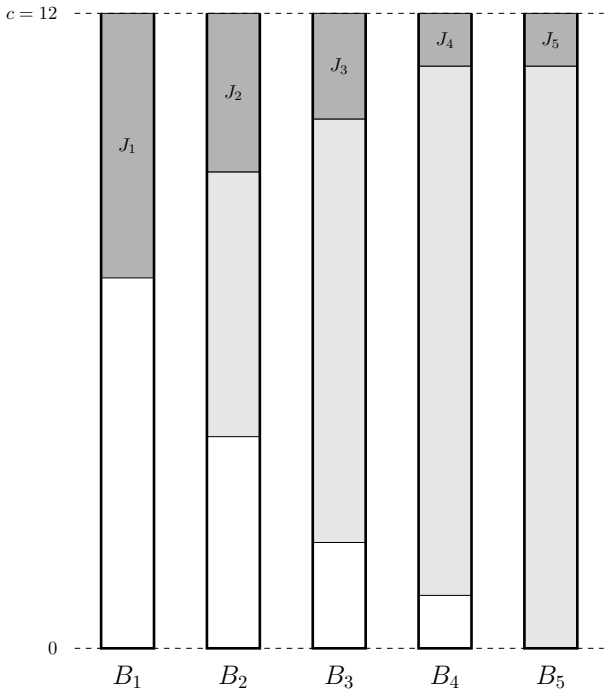


FIGURE 5.6 – Solution donnée par tout algorithme online pour l'exemple

Ce dernier résultat nous invite donc à considérer d'autres approches pour la résolution de ce problème que des méthodes online. Dans la suite de ce chapitre, nous investiguons plusieurs méthodes de la littérature des problèmes de cutting et packing, heuristiques et exactes, basées sur la programmation linéaire en nombre entiers afin de résoudre efficacement le problème.

5.5 Formulation polynomiale

Dans un premier temps, nous investiguons la formulation polynomiale du Bin-Packing, classiquement attribuée à Kantorovich (1960). On suppose que les items sont triés selon l'ordre décroissant de $t_i - w_i$, conformément à la proposition 5.4.2. Deux familles de variables de décision sont considérées.

$$x_i^b = \begin{cases} 1 & \text{si } J_i \text{ affecté à la boîte } B_b \\ 0 & \text{sinon.} \end{cases}$$

$$y^b = \begin{cases} 1 & \text{si la boîte } B_b \text{ est ouverte} \\ 0 & \text{sinon.} \end{cases}$$

Le modèle s'écrit alors de la manière suivante :

$$
\begin{aligned}
min \quad & \sum_{b \in \mathcal{N}} y^b & & & & (5.14)\\
s.t. \quad & \sum_{i \in \mathcal{N}} x_i^b \leq n y^b & \forall b \in \mathcal{N} & & & (5.15)\\
& \sum_{k=1}^{i-1} w_k x_k^b + t_i x_i^b \leq c & \forall i \in \mathcal{N} & \forall b \in \mathcal{N} & & (5.16)\\
& \sum_{b \in \mathcal{N}} x_i^b = 1 & \forall i \in \mathcal{N} & & & (5.17)\\
& y^b \in \{0,1\} & \forall b \in \mathcal{N} & & & (5.18)\\
& x_i^b \in \{0,1\} & \forall i \in \mathcal{N} & \forall b \in \mathcal{N} & & (5.19)
\end{aligned}
$$

L'équation (5.14) définit l'objectif à minimiser comme le nombre de boîtes ouvertes. L'équation (5.15) impose que les items ne peuvent pas être assignés à des boîtes fermées. L'équation (5.16) correspond à la contrainte de seuil et implique également la contrainte de capacité. Elle impose que pour tout remplissage partiel d'une boîte, la capacité restante est supérieure au seuil du prochain item à ranger dans la boîte. Enfin, la contrainte (5.17) impose que chaque item est assigné à exactement une boîte.

Ce modèle implique $O(n^2)$ variables et $O(n^2)$ contraintes. Néanmoins, il est connu que pour le modèle polynomial du Bin-Packing classique, on peut observer qu'en considérant les instances avec des items tels que $w_i = c/2 + \varepsilon$, la borne inférieure fournie par la relaxation linéaire tend vers 1/2 de l'optimal (Martello & Toth, 1990). Notre modèle polynomial étant une généralisation de ce modèle, on obtient exactement le même résultat en posant $t_i = w_i = c/2+\varepsilon$. En raison de cette mauvaise relaxation linéaire, il est attendu que ce modèle soit assez mauvais en pratique pour la résolution directe, et on lui préférera les méthodes basées sur les modèles exponentiels et pseudo-polynomiaux présentées ci-après. Nous revenons sur les performances en pratique de ce modèle dans la section consacrée au résultats expérimentaux.

5.6 Formulation exponentielle et Génération de colonnes

Les formulations exponentielles "set-covering" et "set-partitioning" ont été très largement étudiées dans la littérature du BPP/CSP et sont utilisées dans de nombreuses méthodes de résolution (Belov & Scheithauer, 2006 ; Delorme et al., 2016 ; Gilmore & Gomory, 1961 ; Vance, 1998). En raison de leur nombre exponentiel de variables, elles ne sont pas utilisées pour la résolution directe par un solveur. En revanche, contrairement à la formulation polynomiale, ces formulations ont une très bonne relaxation linéaire. Pour le BPP et le CSP, il a été conjecturé que l'écart entre l'arrondi de la borne inférieure fournie par la relaxation linéaire et l'optimum est au plus égal à 1. Cette conjecture est connue sous le nom de MIRUP (Modified Integer Round-Up Property) et est encore ouverte aujourd'hui (Delorme et al., 2016 ; Scheithauer & Terno, 1997). L'optimum de la relaxation linéaire de ce modèle, malgré le nombre exponentiel de variables, peut être calculé efficacement par génération de colonnes. Dans cette section, nous proposons une adaptation de la formulation "set-partitioning" et de la méthode de génération de colonnes proposée par Gilmore et Gomory (1961).

5.6.1 Formulation exponentielle

Dans la formulation "set-partitioning", chaque variable décrit une combinaison réalisable d'items au sein d'une même boîte parmi l'ensemble des combinaisons possibles $\mathcal{B}$, appelées colonnes.

$$y^b = \begin{cases} 1 & \text{si la colonne } B_b \text{ est selectionnée} \\ 0 & \text{sinon.} \end{cases}$$

L'objectif est alors de trouver un plus petit sous-ensemble de colonnes tel que chaque item apparaisse dans exactement un sous-ensemble. Pour décrire une combinaison d'items dans une boîte, nous introduisons la quantité a_i^b, égale à 1 si l'item i appartient à la colonne b et 0 sinon. Le modèle s'écrit alors de la manière suivante :

$$min \quad \sum_{b \in \mathcal{B}} y^b \tag{5.20}$$

$$s.t. \quad \sum_{b \in \mathcal{B}} a_i^b y^b = 1 \qquad \forall i \in \mathcal{N} \tag{5.21}$$

$$y^b \in \{0, 1\} \qquad \forall b \in \mathcal{B} \tag{5.22}$$

L'équation (5.20) définit l'objectif comme la minimisation du nombre de colonnes sélectionnées dans la solution, ce qui revient à minimiser le nombre de boîtes ouvertes. La contrainte (5.21) impose que la solution soit une partition de l'ensemble des items. Au total, le modèle contient $O(|\mathcal{B}|) = O(2^n)$ variables et $O(n)$ contraintes.

Dans son écriture mathématique, ce modèle est rigoureusement équivalent à celui de Gilmore et Gomory (1961). Cependant, l'ensemble des colonnes est plus restreint : en plus de la contrainte de capacité, les contraintes de seuils doivent être également respectées, ce qui élimine une partie des solutions réalisables du BPP classique.

5.6.2 Schéma de génération de colonnes

La génération de colonnes permet de calculer l'optimum de la relaxation linéaire du modèle exponentiel en ne considérant qu'un sous-ensemble des variables du modèle, généré de manière itérative. Le fonctionnement général des algorithmes de générations de colonnes a été introduit dans le chapitre 1, nous présentons ici l'application au cas de la relaxation linéaire du BPS. L'intégralité des variables de décision est donc relaxée et l'ensemble $\mathcal{B}$ est remplacé par $\mathcal{B}' \subseteq \mathcal{B}$.

$$min \quad \sum_{b \in \mathcal{B}'} y^b \tag{5.23}$$

$$s.t. \quad \sum_{b \in \mathcal{B}'} a_i^b y^b = 1 \qquad \forall i \in \mathcal{N} \tag{5.24}$$

$$0 \leq y^b \leq 1 \qquad \forall b \in \mathcal{B}' \tag{5.25}$$

Ce nouveau modèle est appelé Problème Maître Restreint (RMP). La procédure de génération de colonnes est détaillée sur la Figure 5.7.

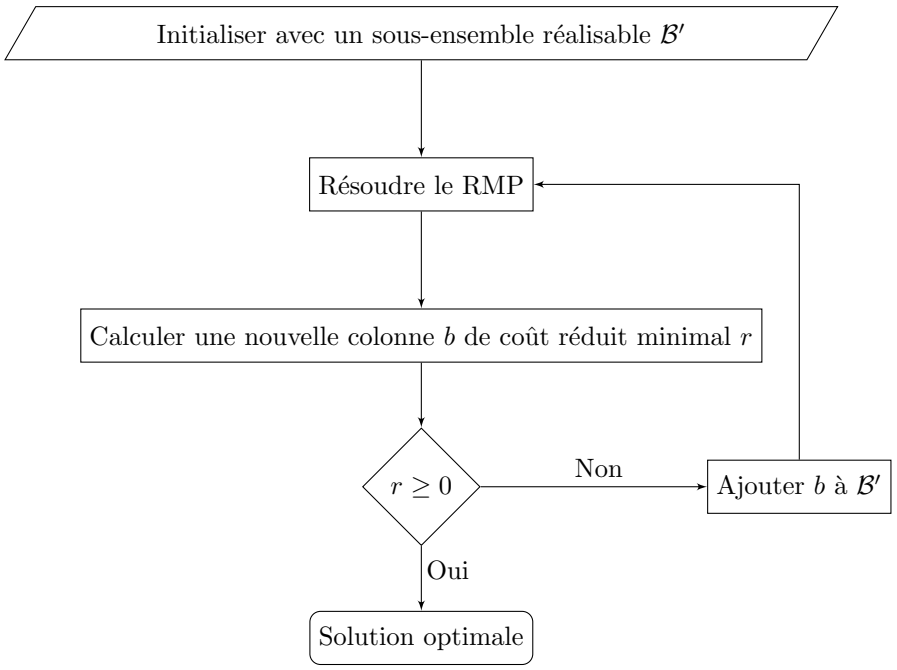


FIGURE 5.7 – Procédure de génération de colonnes

La résolution est initialisée avec un ensemble de colonnes qui permet de construire au moins une solution réalisable. Ici, nous avons choisi l'ensemble des colonnes à un seul item, qui garantit l'existence d'une solution réalisable entière en sélectionnant toutes les colonnes. Ensuite, le RMP est résolu et une nouvelle colonne de coût réduit minimal est calculée. Intuitivement, il s'agit de la colonne qui diminue le plus la valeur de l'objectif si elle est rajoutée à $\mathcal{B}'$. Si le coût réduit est positif ou nul, alors il n'existe pas de colonne améliorante et la solution donnée par le RMP est optimale, sinon on réitère en ajoutant la nouvelle colonne à $\mathcal{B}'$.

En notant v_i la valeur de la variable duale correspondant à la contrainte (5.21) d'appartenance de l'item J_i à la partition, le coût réduit r d'une colonne $b \in \mathcal{B}$ s'exprime de la manière suivante :

$$r = 1 - \sum_{i \in \mathcal{N}} v_i a_i^b \tag{5.26}$$

Calculer une nouvelle colonne de coût réduit minimal revient à trouver un remplissage d'une boîte qui minimise le coût réduit, tel que les contraintes de capacité et de seuils soient respectées. Ce sous-problème est habituellement appelé "problème de pricing". Il s'agit d'une variante du problème du sac à dos avec une contrainte de seuil, où l'utilité des items correspond aux valeurs des variables duales.

Posons l'ensemble de variables de décision suivant :

$$x_i = \begin{cases} 1 & \text{si l'item } J_i \text{ est selectionné} \\ 0 & \text{sinon.} \end{cases}$$

Ce sous-problème peut s'écrire comme un programme linéaire en nombres entiers :

$$max \quad \sum_{i \in \mathcal{N}} v_i x_i \tag{5.27}$$

$$s.t. \quad \sum_{i=1}^{k-1} w_i x_i + t_k x_k \leq c \qquad \forall k \in \{2 \dots n\} \tag{5.28}$$

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{N} \tag{5.29}$$

Minimiser le coût réduit revient à maximiser l'utilité du sac à dos, comme exprimé dans l'équation de l'objectif (5.27). L'équation (5.28) impose que la sélection d'items vérifie les contraintes de seuils et de capacité.

Ce problème étant une variante du problème de sac-à-dos binaire, il peut être également modélisé comme un programme dynamique, avec une résolution en temps pseudo-polynomial. En supposant les items triés par ordre décroissant $t_i - w_i$, posons $V(k, p)$ la valeur de l'objectif pour l'optimum du sous-problème où seuls les k premiers items sont considérés, avec une capacité restreinte p pour la contrainte de capacité. Le programme dynamique s'écrit de la manière suivante.

$$V(k,p) = \begin{cases} 0 & \text{si } k = 0 \\ V(k-1,p) & \text{si } k \geq 1 \text{ et } w_k > p \\ max\{V(k-1,p), v_k + V(k-1, min(p - w_k, c - t_{k)})\} & \text{sinon} \end{cases}$$

Pour le premier cas, comme $k = 0$, il n'y a aucun item restant à placer dans la boîte, par conséquent la valeur de l'objectif est 0. Dans le second cas, la contrainte de capacité empêche de placer le k-ième item, la seule décision possible est de ne pas prendre l'item et de passer à l'item suivant, la valeur totale de l'objectif reste inchangée.

Enfin, le dernier cas correspond au cas général. Comme, pour le programme dynamique du sac à dos classique, il y a deux possibilités. Soit on choisit de ne pas prendre l'item, ce qui revient au cas précédent, soit on choisit de prendre l'item. Dans ce dernier cas, il y a deux possibilités, illustrées sur la Figure 5.8.

Si le seuil t_k de l'item J_k est plus petit que la portion de sac à dos indisponible après avoir ajouté l'item J_k, alors c'est la contrainte de capacité qui est active et la meilleure décision consiste à placer l'item au point le plus profond du sac. Cette situation correspond au Cas 1 de la Figure 5.8. La capacité restante est alors $p - w_k$. Dans le cas contraire, c'est la contrainte de seuil qui est active et l'item est placé au point le profond possible du sac, tel que la contrainte de seuil soit respectée, soit $c - t_k$. Cela signifie qu'une portion du sac de taille $p - c - w_k + t_k$ est perdue et ne sera pas occupée. Cette situation correspond au Cas 2 de la Figure 5.8.

Le calcul de $V(n, c)$ nécessite de calculer les valeurs de $V(k, p)$ pour $0 \leq k \leq n$ et $0 \leq p \leq c$, et chacune de ces valeurs se calcule en un temps constant. Ce programme dynamique permet

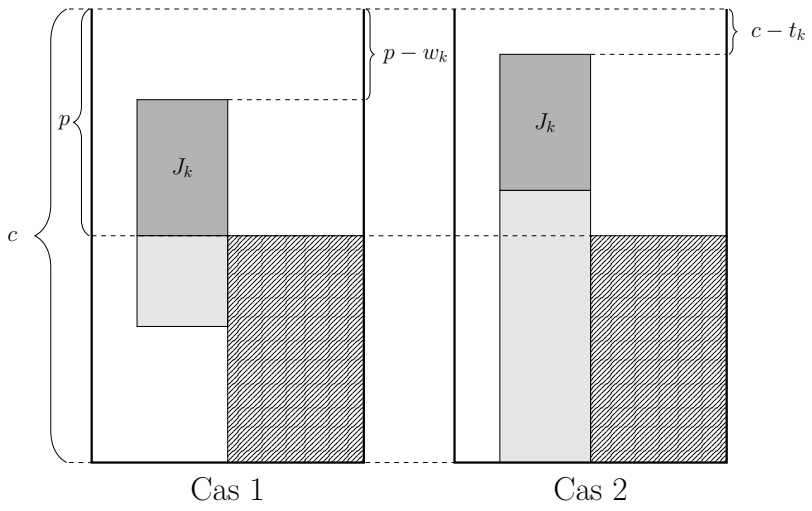


FIGURE 5.8 – Deux cas du programme dynamique

donc de résoudre le problème de pricing en temps pseudo-polynomial $O(nc)$. L'équivalence avec le programme linéaire est montrée ci-après. Notons $f(k,p)$, la valeur optimale du PLNE de pricing avec restriction aux k premiers items et une capacité restreinte p pour la contrainte de capacité.

$$f(k,p) = max\Bigg\{\sum_{i=1}^{k} v_i x_i \Bigg| \sum_{i=1}^{k} w_i x_i \leq p, \quad \forall l \in \{2 \dots k\}, \sum_{i=1}^{l-1} w_i x_i + t_l x_l \leq c, \quad \forall i \in \{1 \dots k\}, x_i \in \{0,1\}\Bigg\}$$

Notons que $f(n,c)$ correspond à la solution optimale du problème de pricing. On veut montrer la proposition suivante :

Proposition 5.6.1. $f(n,c) = V(n,c)$.

Pour ce faire, prouvons par induction la propriété suivante :

$$\forall k \in \{1 \dots n\},\ \forall p \in \{1 \dots c\},\ P(k,p) : f(k,p) = V(k,p) \tag{5.30}$$

Démonstration. L'initialisation est faite pour les cas $k = 0$, $0 \leq p \leq c$, où il est facile de vérifier que $f(0,p) = V(0,p) = 0$.

Prouvons que $\forall k \in \{1 \dots n\}, \forall p \in \{0 \dots c\}, \bigwedge_{\substack{0 \leq k' < k \\ 0 \leq k' \leq c}} P(k',p') \implies P(k,p)$

Pour tout $1 \leq k \leq n$ et tout $0 \leq p \leq c$, il est possible de réorganiser les termes de l'objectif et des contraintes :

$$f(k,p) = max\Bigg\{\sum_{i=1}^{k-1} v_i x_i + v_k x_k \Bigg| \sum_{i=1}^{k-1} w_i x_i \leq p - w_k x_k,$$
$$\forall l \in \{2 \dots k-1\}, \sum_{i=1}^{l-1} w_i x_i + t_l x_l \leq c,$$
$$\sum_{i=1}^{k-1} w_i x_i \leq c - t_k x_k,$$
$$\forall i \in \{1 \dots k\}, x_i \in \{0,1\}\Bigg\}$$

On réalise alors une disjonction sur $x_k = 0$ et $x_k = 1$, et notons respectivement A_0 et A_1 les valeurs d'objectif correspondantes. $f(k,p) = max(A_0, A_1)$, où :

$$A_0 = max\Bigg\{\sum_{i=1}^{k-1} v_i x_i \Bigg| \sum_{i=1}^{k-1} w_i x_i \leq p,$$
$$\forall l \in \{2 \dots k-1\}, \sum_{i=1}^{l-1} w_i x_i + t_l x_l \leq c,$$
$$\sum_{i=1}^{k-1} w_i x_i \leq c,$$
$$\forall i \in \{1 \dots k-1\}, x_i \in \{0,1\}\Bigg\}$$

Puisque $p \leq c$, le troisième jeu de contraintes peut être supprimé, et $A_0 = f(k-1,p)$. En appliquant l'hypothèse d'induction, $A_0 = V(k-1,p)$.

$$A_1 = v_k + max\Bigg\{\sum_{i=1}^{k-1} v_i x_i \Bigg| \sum_{i=1}^{k-1} w_i x_i \leq p - w_k,$$
$$\forall l \in \{2 \dots k-1\}, \sum_{i=1}^{l-1} w_i x_i + t_l x_l \leq c,$$
$$\sum_{i=1}^{k-1} w_i x_i \leq c - t_k,$$
$$\forall c \in \{1 \dots k-1\}, x_i \in \{0,1\}\Bigg\}$$

Dans ce cas il y a deux possibilités : si $p < w_k$, par positivité de $\sum_{i=1}^{k-1} w_i x_i$, la première contrainte ne peut pas être respectée et $A_1 = max(\emptyset) = -\infty$. Sinon, en combinant le premier

et le troisième jeu de contraintes, on obtient $A_1 = f(k-1, min(p-w_k, c-t_k))$. Par hypothèse d'induction $A_1 = f(k-1, min(p-w_k, c-t_k))$.

Finalement, pour tout $1 \leq k \leq n$ et tout $0 \leq p \leq c$:

$$
\begin{aligned}
f(k,p) &= max(A_0, A_1) \\
&= \begin{cases} max(V(k-1,p), -\infty) = V(k-1,p) & \text{si } w_k > p \\ max\{V(k-1,p), v_k + V(k-1, min(p-w_k, c-t_k))\} & \text{sinon} \end{cases} \\
&= V(k,p)
\end{aligned}
$$

On a donc montré la propriété $P(k,p)$ pour toute valeur k et p, on en déduit donc la proposition. On a donc bien l'équivalence entre le programme linéaire et le programme dynamique.

□

5.6.3 Reconstruction d'une solution entière

La génération de colonnes permet de résoudre la relaxation linéaire du modèle exponentiel. Elle fournit donc une borne inférieure sur l'optimum du PLNE. Si la solution renvoyée est entière, alors cela signifie que l'optimum de la relaxation coïncide avec l'optimum du PLNE. Dans le cas contraire, une étape supplémentaire est nécessaire pour obtenir une solution réalisable. Les différentes méthodes de reconstruction d'une solution réalisable ont été recensées dans un article récent de Sadykov et al. (2019).

Ici, nous avons opté la méthode de l'heuristique du problème maître, également connue sous le nom de price-and-branch, qui consiste à résoudre le RMP en variables entières avec les colonnes précédemment générées. Les colonnes sélectionnées pour l'initialisation de la génération de colonnes garantissent l'existence d'une solution entière réalisable. Les performances de la méthode seront discutées dans la section 5.8.

5.7 Formulation arc-flot

Les méthodes arc-flot consistent à formuler le problème comme un problème de flot dans un graphe orienté, résolu par programmation linéaire en nombres entiers. En comparaison avec d'autres méthodes basées sur la programmation linéaire, les méthodes arc-flot sont en général compétitives et faciles d'implémentation puisqu'il est possible de les implémenter directement dans un solveur de programmation linéaire (De Lima et al., 2022). Dans cette section, la formulation arc-flot proposée par Valério de Carvalho (2002) est généralisée pour prendre en compte les contraintes de seuil du problème. Les autres méthodes arc-flot existantes dans la littérature (Brandão & Pedroso, 2016 ; Delorme & Iori, 2020) utilisent des brisures de symétries sur l'ordre de remplissage des boîtes, qui font qu'elles ne sont pas directement adaptables au problème.

5.7.1 Construction du graphe

Un remplissage d'une boîte peut être modélisé comme un chemin dans un graphe orienté acyclique : posons $G = (V, A)$ le graphe tel que chaque sommet représente une capacité restante dans la boîte $V = \{0 \dots c\}$ et tel que chaque arc représente un emplacement du sac pouvant être occupé par un item. On reprend l'exemple de la section, avec une capacité $c = 4$ et les caractéristiques suivantes :

Item	J_1	J_2	J_3	J_4
w_i	2	1	1	2
t_i	3	3	2	3

Pour cet exemple, le remplissage de la boite B_2 sur la Figure 5.1 est représenté par le chemin formé par les arcs $(4, 3)$ et $(3, 1)$ dans un graphe à 5 sommets $\{0, 1, 2, 3, 4\}$.

Toute paire de sommets du graphe est reliée par un arc s'il existe un item dont le poids correspond à la différence de capacité entre les deux sommets. Sur l'exemple, il y aura donc les arcs correspondants à des items de poids 1 et 2. De plus, pour assurer la conservation du flot, des arcs fictifs reliant chaque sommet, hormis la source c et le puits 0, au puits sont ajoutés. Formellement, l'ensemble des arcs et défini par :

$$A = \left\{(p, q) \in V^2 | \exists k \in \mathcal{N},\ p - w_k = q\right\} \cup \{(p, 0) | p \in \{1 \dots c - 1\}\} \tag{5.31}$$

Cette construction pour notre exemple est représenté sur la Figure 5.9.

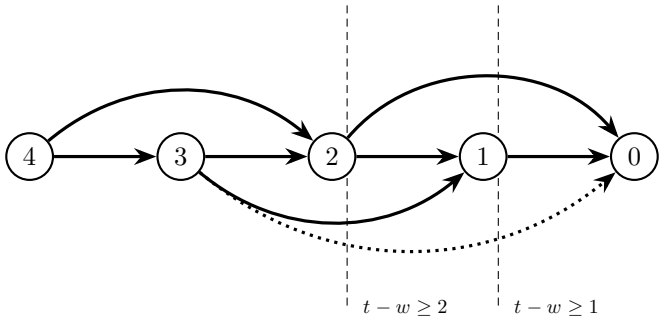


FIGURE 5.9 – Graphe de l'exemple

Pour le BPP classique, en l'absence de la contrainte de seuil, tout emplacement de la bonne taille peut être assigné à un item, indépendamment de son emplacement dans la boîte. Au contraire, dans le cas du BPS, la faisabilité d'une solution dépend aussi de l'emplacement des items dans la boîte. Sur l'exemple de la section 5.2, l'item J_1 ne peut pas être assigné à l'arc $(2, 0)$, car la contrainte de seuil impose que la capacité restante soit au moins 3 au moment de placer l'item. On peut alors représenter une solution du BPS comme un ensemble de chemins correspondant chacun à une boîte réalisable, c'est-à-dire, $(c, 0)$-flot dans le graphe G. L'objectif étant de minimiser le nombre de boîtes, on cherche donc un flot de valeur minimal qui respecte l'ensemble des contraintes du problème.

On regroupe les items en classes d'équivalence d'items de même poids et même seuils. L'ensemble de ces classes d'équivalences est défini de la manière suivante :

$$\mathcal{N}' = \{\gamma = (w_\gamma, t_\gamma) \in \{0 \dots c\}^2 | \exists i \in \mathcal{N},\ w_i = w_\gamma,\ t_i = t_\gamma\} \tag{5.32}$$

Pour chacune de ces classes γ, on définit la quantité suivante :

$$b_\gamma = |\{i \in \mathcal{N} | w_i = w_\gamma,\ t_i \geq t_\gamma\}| \tag{5.33}$$

Intuitivement, cette quantité représente le nombre d'items de poids w_γ pour lesquels la capacité restante doit être au moins t_γ lorsqu'ils sont placés dans une boite. Dans la modélisation arc-flot, cette contrainte impose que pour chaque classe d'équivalence γ, la somme des valeurs du flot à travers chacun des arcs de longueur w_γ du sous-graphe induit par les sommets $\{t_\gamma - w_\gamma \dots c\}$ est supérieure ou égale à b_γ. Les sous-graphes correspondants sont délimités par les pointillés sur la Figure 5.9.

Sur l'exemple, les items J_1 et J_4 sont regroupés au sein de la même classe d'équivalence. On a alors pour la classe d'équivalence $\{J_1, J_4\}$ la valeur $b_1 = |\{J_1, J_4\}| = 2$. Pour la classe $\{J_2\}$, on a $b_2 = |\{J_2\}| = 1$ et pour la classe $\{J_3\}$, on a $b_3 = |\{J_2, J_3\}| = 2$. Dans le sous-graphe induit par les sommets $\{2, 3, 4\}$, la somme de la valeur des flots à travers les arcs de longueur 1 doit être supérieure ou égale à 1 pour l'item J_2. De même, pour le sous-graphe induit par les sommets $\{1, 2, 3, 4\}$, cette valeur pour les arcs de longueur 1 (resp. 2) doit être supérieure ou égale 2 pour les items J_2 et J_3 (resp. J_1 et J_4).

Un arc-flot respectant ces contraintes est représenté sur la Figure 5.10. Une fois un tel arc-flot obtenu, il est possible de reconstruire une solution au problème avec un nombre de boîtes égal à la valeur de l'arc-flot en utilisant l'algorithme de reconstruction présenté dans la sous-section 5.7.3.

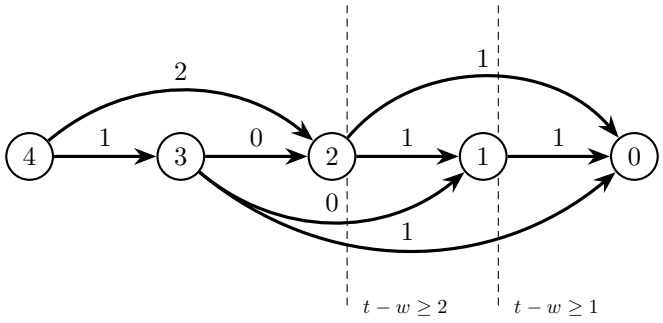


FIGURE 5.10 – Arc-flot obtenu pour l'exemple

5.7.2 Modèle de PLNE

En résumé, on cherche un arc-flot de valeur minimale dans un graphe orienté acyclique tel qu'on puisse assigner à chaque item de l'instance un arc qui satisfait la contrainte de seuil. Ce

problème est modélisé par PLNE, avec deux types de variables : $x_{pq} \in \mathbb{N}$ représentant le flot à travers chaque arc $(p, q) \in A$ et $z \in \mathbb{N}$ représentant la valeur du flot.

$$min \quad z \tag{5.34}$$

$$s.t. : \quad \sum_{(p,r)\in A} x_{pr} - \sum_{(r,q)\in A} x_{rq} = \begin{cases} -z & \text{si } r = c \\ z & \text{si } r = 0 \\ 0 & \forall r \in \{1 \dots c-1\} \end{cases} \tag{5.35}$$

$$\sum_{\substack{(p,p+w_\gamma)\in A \\ p \geq t_\gamma}} x_{p,p+w_\gamma} \geq b_\gamma \qquad \forall \gamma \in \mathcal{N}' \tag{5.36}$$

$$x_{pq} \in \mathbb{N} \qquad \forall (p, q) \in A \tag{5.37}$$

$$z \geq 0 \tag{5.38}$$

L'objectif à minimiser (5.34) est la valeur du flot z. La contrainte (5.35) est une contrainte de conservation du flot classique : pour chaque sommet du graphe, la différence entre le degré sortant et le degré entrant du flot doit être nulle, à l'exception de la source c et du puits 0 pour lesquels cette différence est égale à la valeur du flot. La contrainte (5.36) impose que la somme des valeurs du flot à travers les arcs de longueur w_γ dans le sous-graphe induit par les sommets $\{t_\gamma - w_\gamma \dots c\}$ est supérieure ou égale à b_γ afin de garantir la satisfaction de la contrainte de seuil.

La particularité de ce modèle réside dans le nombre pseudo-polynomial de variables $O(c^2)$ et de contraintes $O(n + c)$. Autrement dit, le nombre de variables du modèle ne dépend pas du nombre d'items, mais uniquement de la capacité des boîtes.

5.7.3 Algorithme de reconstruction

Le modèle de programmation linéaire en nombres entiers présenté au paragraphe précédent renvoie une solution sous la forme d'un flot dans un graphe orienté correspondant à une solution réalisable du problème initial. Il est donc nécessaire de passer par une étape de post-processing afin de reconstruire la solution correspondante. Il s'agit d'une procédure gloutonne détaillée dans l'algorithme 1.

Elle consiste à décomposer le flot en chemins, en utilisant les propriétés de décomposition des flots (Ahuja et al., 1988). Les items sont ensuite triés par ordre décroissant $t_i - w_i$, c'est-à-dire du plus difficile à placer au plus facile et assignés au premier arc libre de longueur adéquate tel que les contraintes de seuils soient respectées. À la fin de la procédure, tous les items se retrouvent assignés à un chemin, correspondant à une boîte réalisable.

La décomposition et une assignation correspondantes pour le flot réalisable de la Figure 5.10 sont représentées sur la Figure 5.11. L'algorithme de décomposition de flots en chemin consiste à chercher successivement des $(c, 0)$-chemins tels que la valeur du flot à travers le chemin soit non nuls, puis à diminuer la valeur du flot le long du chemin. Sur l'exemple, en partant du sommet $c = 4$, on sélectionne l'arc $(4, 2)$ dont la valeur est 2, puis l'arc $(2, 0)$ dont la valeur est 1. La valeur du flot à travers ce chemin C_1 est $1 > 0$, on l'extrait et on diminue la valeur du flot de 1 le long du chemin. En répétant le procédé, on extrait successivement les chemins $C_2 = (4, 2), (2, 1), (1, 0)$ et $C_3 = (4, 3), (3, 0)$. Les arcs $(4, 2)$ et $(4, 3)$ forment alors

Algorithme 1 Reconstruction de la solution

Entrée $F = (x_a)_{a \in A}, z$ solution du modèle arc-flot
Entrée L liste d'items triés par ordre décroissant $t_i - w_i$
Sortie Solution réalisable (ensemble de boîtes) S

$S \leftarrow \bigcup_{m=1}^{z} B_m$ où $\forall m \in \{1 \dots z\}, B_m = \emptyset$
Transformer F en un ensemble de z chemins P en utilisant les propriétés de décomposition des flots
pour Chaque item J_i dans L **faire**
    pour Chaque chemin $m \in P$ **faire**
        si m contient un arc non-affecté $a = (p, p - w_i)$ tel que $p \geq t_i$ **alors**
            Affecter l'arc a à l'item J_i
            Ajouter J_i à la position correspondante dans la boîte B_m
            Passer à l'item suivant
        fin si
    fin pour
fin pour
Renvoyer S

une $(4, 0)$-coupe de capacité nulle, ce qui garantit qu'il n'est plus possible d'extraire d'autres chemins. On retrouve donc bien 3 chemins pour un flot de valeur initiale 3. On construit donc trois boîtes B_1, B_2 et B_3 correspondant aux chemins respectifs C_1, C_2 et C_3. On trie ensuite les items selon l'ordre décroissant $t_i - w_i$, ce qui donne la liste ordonnée suivante $\{J_2, J_1, J_3, J_4\}$. On commence alors par placer l'item J_2. Dans le chemin C_1, tous les arcs sont de longueur 2 et J_2 a un poids $w_2 = 1$ et donc aucun arc ne correspond. Dans le chemin C_2, les arcs $(2, 1)$ et $(1, 0)$ correspondent pour le poids, mais ne respectent pas la contrainte de seuil de J_2, car $t_2 - w_2 = 2$. Enfin dans le chemin C_3, l'arc $(4, 3)$ satisfait toutes les contraintes. On assigne donc J_2 à la boîte B_3. On répète le procédé jusqu'à ce que tous les items soient assignés à un arc d'un des chemins, et par conséquent à un emplacement dans une boîte. Finalement, on arrive à l'assignation suivante $B_1 = (J_1)$, $B_2 = (J_4, J_3)$, $B_3 = (J_2)$. En raison de la contrainte de conservation du flot, certains arcs restent non-affectés à la fin de la procédure. Néanmoins, tous les items sont correctement assignés et la solution obtenue est donc réalisable. Cela est garanti par le fait que les items sont affectés successivement selon l'ordre décroissant $t_i - w_i$ et par construction de l'arc-flot.

5.8 Résultats expérimentaux

Les trois méthodes de résolution présentées précédemment ont été testées et comparées sur un jeu d'instance construit à partir d'instances de la littérature. L'ensemble des expérimentations a été réalisé sur un ordinateur portable doté d'un processeur Intel Core i7-10610U CPU @ 1.80GHz x 8 et 16 Go de mémoire RAM, tournant sous Ubuntu 22.04.1 LTS. L'implémentation a été effectuée en C++17 avec l'API IBM ILOG CPLEX 22.1. Sauf mention explicite du contraire, le temps limite par expérimentation est fixé à 1 minute.

Deux benchmarks ont été construits à partir de benchmarks classiques du BPP. Le premier

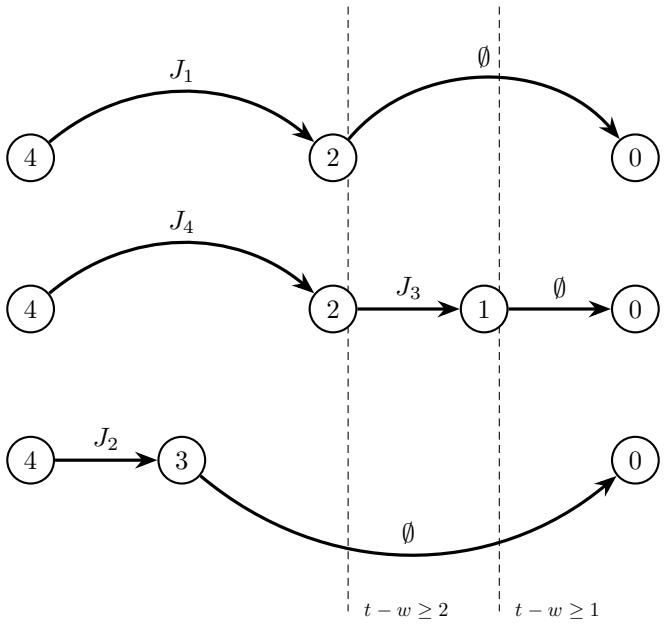


FIGURE 5.11 – Décomposition en chemin et assignation des items

benchmark a été construit en ajoutant des seuils aux instances aléatoires de (Delorme et al., 2016). Afin de limiter la combinatoire, nous sélectionnons un sous-ensemble de ces instances, tout en en préservant la diversité. Les instances sélectionnées ont les caractéristiques suivantes :

— $n \in \{50, 100, 200, 300, 400, 500, 750, 1000\}$
— $c \in \{50, 75, 100, 120, 125, 150, 200, 300, 400, 500, 750, 1000\}$
— $w_i \in [0.2c, 0.8c]$

Pour chacune de ces instances du BPP, trois instances du BPS sont construites en utilisant des méthodes de génération de seuils différentes : seuils proportionnels au poids $t_i = min(c, 1.1w_i)$, seuils inversement proportionnels au poids $t_i = max(w_i, 0.75(c - w_i))$, et seuils tirés uniformément dans l'intervalle $[w_i, c]$. Au total, ce benchmark est constitué de 288 instances. Ce premier jeu d'instances permet de comparer nos méthodes sur un benchmark quelconque ne présentant pas de difficultés particulières, avec une grande diversité sur la valeur des différents paramètres, ce qui permet d'évaluer la sensibilité des méthodes à ceux-ci.

Le second benchmark est constitué en appliquant le même procédé de génération de seuils aux instances plus difficiles du benchmark Hard28 de Schoenfield (2002), pour un total de 84 instances avec $n \in [160, 200]$ et $c = 1000$. L'objectif de ce benchmark est de comparer les performances des méthodes sur des instances présélectionnées de la littérature ayant la réputation d'être plus difficiles.

Un dernier benchmark a été réalisé pour tester la sensibilité des méthodes à la multiplicité des items, constitué de 25 instances générées à partir du benchmark de Delorme et al. (2016) avec des seuils aléatoire, $n = 50$ et $c = 500$. Chaque item est répliqué μ_n fois où μ_n est tiré respectivement dans les intervalles $[1, 2], [2, 3], [3, 5], [5, 10], [10, 20]$.

Plusieurs indicateurs, nombre d'instances résolues à l'optimum, temps de calcul moyen

et médian, sur le premier benchmark sont présentés dans les Tables 5.1, 5.2, 5.3 et 5.4. La première colonne représente la valeur du paramètre qui varie, respectivement le nombre d'items n ou la capacité des boîtes c. Les deuxièmes, troisièmes et quatrièmes colonnes représentent respectivement la valeur des indicateurs pour chacune des trois méthodes.

n	**Mod. Poly.**	**Gen. Col.**	**Arc-Flot**
50	31	30	33
100	17	24	33
200	3	18	33
300	0	18	33
400	0	10	33
500	0	13	33
750	0	9	33
1000	0	8	33

TABLE 5.1 – Nombre d'instances résolues à l'optimum en moins d'une minute lorsque n varie. (36 instances par groupe, $c \in [50, 1000]$)

c	**Mod. Poly.**	**Gen. Col.**	**Arc-Flot**
50	5	9	**24**
75	3	11	**24**
100	5	14	**24**
120	3	9	**24**
125	5	10	**24**
150	5	12	**24**
200	3	9	**24**
300	5	10	**24**
400	5	12	**24**
500	5	12	**24**
750	5	11	**24**
1000	4	11	0 (1000)

TABLE 5.2 – Nombre d'instances résolues à l'optimum en moins d'une minute lorsque c varie. (24 instances par groupe, $n \in [50, 1000]$)

Sur l'ensemble du benchmark composé de 288 instances, la formulation polynomiale a permis de trouver des solutions réalisables pour 145 instances et l'optimum pour seulement 51 d'entre-elles. À l'opposé, on constate que les deux autres méthodes sont beaucoup plus efficaces. La génération de colonnes couplée à l'heuristique du problème maître restreint fournit des solutions réalisables pour l'ensemble des instances. De plus, toutes ces solutions sont proches de l'optimum, comme cela peut être constaté dans la table 5.5. L'optimalité est prouvée pour 130 d'entre-elles, 228 ont un gap absolu inférieur à 1, c'est-à-dire que la solution obtenue contient au plus une boîte de plus que l'optimum et le plus grand gap absolu trouvé

	Mod. Poly.		Gen. Col.		Arc-Flot	
n	moyenne	médiane	moyenne	médiane	moyenne	médiane
50	6.46	0.82	0.05	0.03	9.65	0.38
100	36.54	60.00	0.07	0.07	9.84	0.41
200	58.55	60.00	0.18	0.17	10.17	0.46
300	60.00	60.00	0.37	0.37	10.41	0.49
400	60.00	60.00	0.64	0.62	10.61	0.50
500	60.00	60.00	1.07	1.02	10.69	0.50
750	60.00	60.00	1.07	1.02	11.28	0.51
1000	60.00	60.00	9.15	8.24	11.80	0.50

TABLE 5.3 – Temps de calcul moyen et médian (sec) quand n varie. (temps limite 60 sec) (36 instances par groupe, $c \in [50, 1000]$)

	Mod. Poly.		Gen. Col.		Arc-Flot	
c	moyenne	médiane	moyenne	médiane	moyenne	médiane
50	50.28	60.00	2.09	0.50	0.03	0.03
75	51.14	60.00	1.95	0.45	0.07	0.07
100	48.01	60.00	1.92	0.50	0.13	0.12
120	53.19	60.00	1.87	0.52	0.19	0.19
125	47.64	60.00	1.80	0.52	0.20	0.20
150	48.78	60.00	1.87	0.40	0.31	0.30
200	53.03	60.00	1.66	0.48	0.69	0.67
300	49.48	60.00	1.45	0.44	2.42	2.43
400	50.50	60.00	2.00	0.49	6.21	6.02
500	50.39	60.00	1.40	0.49	12.28	11.87
750	50.98	60.00	1.81	0.55	44.11	43.55
1000	50.38	60.00	1.75	0.61	60.00	60.00

TABLE 5.4 – Temps de calcul moyen et médian (sec) quand c varie. (temps limite 60 sec) (24 instances par groupe, $n \in [50, 1000]$)

est 5. Enfin, la formulation arc-flot permet de trouver des solutions pour 267 instances, dont 264 optimales.

Nous nous intéressons dans un premier temps à la sensibilité de la performance de résolution des différentes méthodes aux deux paramètres de taille des instances. Les tables 5.1 et 5.3 présentent le nombre d'instances pour lesquelles une solution réalisable a été trouvée et les temps moyens et médian de calcul en fonction du nombre d'items de l'instance pour chaque méthode. Le modèle polynomial parvient à résoudre la plupart des instances pour $n = 50$, mais n'en résout plus aucune à partir de $n = 300$ et utilise l'intégralité du temps imparti pour fournir une solution. La génération de colonnes, bien qu'heuristique, fournit des solutions optimales pour la majorité des petites instances $n \leq 300$ avec des temps de calcul inférieurs en

n	**max abs. gap.**
50	2
100	2
200	4
300	2
400	4
500	3
750	5
1000	4

TABLE 5.5 – Gap absolu moyen (nombre de boîtes) pour la génération de colonne quand n varie.(temps limite 60 sec) (36 instances par groupe, $c \in [50, 1000]$)

moyenne de l'ordre de la seconde ou inférieur sauf pour $n = 1000$. De plus on possède de très bonnes garanties pour les solutions non-optimales. Le gap absolu (c'est-à-dire le nombre de boîtes en plus par rapport à la meilleure borne inférieure trouvée) est toujours faible, comme il est possible de l'observer sur la Table 5.5, le plus grand gap absolu ayant été obtenu étant de 5 sur un total de 389 boîtes. Enfin, la méthode arc-flot résout à l'optimum toutes les instances pour chaque valeur de n à l'exception de trois correspondant à la capacité $c = 1000$ comme nous le verrons par la suite. On observe, conformément au fait que le modèle polynomial et la génération de colonnes ont un nombre de variables dépendant du nombre d'items des instances, que les temps de résolutions augmentent avec le nombre d'items, ce qui n'est pas le cas du modèle arc-flot.

Les tables 5.2 et 5.4 présentent la dépendance des performances de résolution au paramètre de capacité des boîtes. Dans ce cas, les performances du modèle polynomial et de la génération de colonnes n'apparaissent que peu impactées, en revanche, les performances du modèle arc-flot sont très impactées par l'augmentation de la capacité. On observe plus particulièrement que le modèle arc-flot parvient à résoudre à l'optimum toutes les instances pour $c \leq 500$, et y parvient en moins d'une seconde en moyenne pour les instances telles que $c \leq 200$. En revanche, pour $c \leq 1000$, le modèle fait intervenir un million de variables, ce qui excède la quantité de mémoire autorisée par défaut par le solveur. Dans ces conditions expérimentales, notre méthode arc-flot n'est pas capable de résoudre ces instances. Les résultats obtenus sont également en conformité avec le fait que le modèle arc-flot est le seul qui ait un nombre de variables dépendant de la capacité.

Les résultats concernant l'impact de la multiplicité sur les performances de résolution sont présentés dans la table 5.6. Le modèle polynomial ne parvient plus à fournir de solutions réalisables quand la multiplicité dépasse 3. La génération de colonnes parvient toujours à résoudre les instances lorsque la multiplicité est importante, mais le temps de calcul et le gap absolu augmente. La méthode arc-flot toutefois n'est pas du tout impacté par l'augmentation de la multiplicité, ce qui cohérent puisque les items identiques sont regroupés dans des classes d'équivalence avant la résolution du modèle de programmation linéaire en nombres entiers.

Enfin, les modèles ont été testés sur le jeu de 84 instances construit à partir des instances difficiles Hard28 de Schoenfield, avec un temps limite relevé à 600 secondes. Pour chacun des modèles, plusieurs métriques sont présentées dans la table 5.7 : nombre d'instances pour

μ	**Mod. Poly.**	**Gen. Col.**	**Arc-Flot**
$[1, 2]$	60.00 (5)	0.03 (0)	10.74 (0)
$[2, 3]$	60.00 (10)	0.07 (0)	9.00 (0)
$[3, 5]$	-	0.13 (1)	9.93 (0)
$[5, 10]$	-	0.44 (1)	9.66 (0)
$[10, 20]$	-	2.55 (1)	10.07 (0)

TABLE 5.6 – Temps de calcul moyen en sec. (gap absolu moyen) quand μ (multiplicité) varie. (temps limite 60 sec) (5 instances par groupe, $n \in [50, 1000]$, $c = 500$)

lesquelles une solution réalisable a été trouvée, nombre d'instances fermées, temps de calcul moyen et gap absolu moyen. Ces résultats confirment que le modèle polynomial est peu efficace et présente peu d'intérêt pour la résolution en pratique, tandis que la génération de colonnes permet de trouver rapidement des bonnes solutions. La méthode arc-flot s'avère excellente sur ces instances et parvient à obtenir l'optimum dans 80 des 84 cas avec un temps de calcul moyen de l'ordre de deux minutes.

	Mod. Poly.	**Gen. Col.**	**Arc-Flot**
nb. sol. real.	**84**	**84**	**84**
nb. sol. opt.	12	34	80
tps calc. moy. (sec.)	531.57	83.33	136.96
gap abs. max. (nb. boîtes)	70	5	1

TABLE 5.7 – Plusieurs métriques pour le benchmark Hard28 (temps limite 600 sec) (84 instances au total)

Pour conclure, dans l'ensemble la méthode de génération de colonnes couplée à l'heuristique du problème maître restreint et le modèle arc-flot surpassent très largement le modèle polynomial. Bien qu'heuristique, la méthode de génération de colonnes est très efficace et permet de trouver des solutions optimales ou proches de l'optimal pour un très grand nombre d'instances. Elle est peu sensible aux variations de capacité et est donc particulièrement adéquate pour les instances avec grandes capacités. Elle est toutefois assez complexe sur le plan technique, et plus difficile à implémenter. La méthode arc-flot, quant à elle, est très efficace tant que la capacité n'est pas trop élevée. Elle présente l'avantage de ne pas être sensible au nombre d'items et est donc très adaptée pour des instances présentant des grands nombres d'items ou avec une forte multiplicité.

5.9 Conclusion

Dans ce chapitre, nous avons approfondi l'étude du cas de la minimisation du délai maximum C_{max} du problème d'ordonnancement présenté dans le chapitre 4 dans le cas où l'exécution des tâches est conditionnée par le respect de seuils d'acceptabilité sur le niveau de santé de l'équipement. Nous revenons sur les liens avec le BPP, et traduisons cette nouvelle

contrainte dans ce formalisme. Nous proposons alors l'adaptation de trois méthodes de l'état de l'art basées sur la programmation linéaire en nombres entiers en exploitant une règle de dominance sur l'ordre des items dans les boîtes. Les trois méthodes ont été testées, comparées et évaluées sur un jeu d'instance dérivé de la littérature. Dans l'ensemble, il apparait que la génération de colonnes et la méthode arc-flot sont très efficaces. Plus précisément, la génération de colonnes, du fait de son insensibilité à la capacité, est indiquée pour les cas où les capacités sont grandes, mais où le nombre d'items reste raisonnable, tandis que la méthode arc-flot est insensible au nombre d'items et à leur multiplicité, et est donc particulièrement compétitive sur les instances ayant un nombre très important d'items tant que la capacité des boîtes est limitée. Il est raisonnable de supposer que les capacités sont faibles dans notre cas d'étude, puisque cela correspond à la granularité de l'indice de santé, qui n'est a priori pas très élevée, ce qui rend la méthode arc-flot d'autant plus intéressante.

Résumé des contributions et perspectives

Principales contributions Dans cette partie, nous avons introduit un nouveau problème d'ordonnancement conjoint de la production et de la maintenance en présence d'un indicateur de santé des composants d'un équipement. Dans un premier temps, nous avons conduit une étude autour des caractéristiques du problèmes, de ses différentes variantes, de ses contraintes et de ses objectifs. Une cartographie de la complexité des différentes variantes et leur modélisation par la programmation linéaire en nombres entiers sont proposées. Nous revenons ensuite plus en détails sur les deux objectifs fondamentaux de la théorie de l'ordonnancement : minimisation du délai maximum et minimisation du délai moyen. Pour ces sous-problèmes, nous présentons différents résultats autour de la structure des solutions optimales, l'approximabilité, la performance des algorithmes on-line et de la résolution par programmation dynamique.

Nous avons ensuite étudié le cas particulier de la minimisation du temps de complétion du planning en présence de seuils et avec maintenance parfaite. Ce problème est une variante du problème de bin-packing, faisant intervenir une contrainte originale de seuil qui contraint l'ordre de remplissage des boîtes. Trois méthodes basées sur la programmation linéaire en nombres entiers sont proposées : une adaptation de la formulation de Kantorovich, un algorithme de génération de colonnes basée sur la formulation exponentielle "set-partitioning" et une modélisation de type arc-flot. Les méthodes ont été testées et comparées sur un benchmark adapté de benchmarks de la littérature du bin-packing et de cutting stock. Les résultats ont permis d'établir que selon la nature des instances, il est préférable d'utiliser soit la génération de colonnes soit le modèle arc-flot pour la résolution pratique du problème.

Nous présentons ci-dessous quelques perspectives de recherche autour du problème présenté dans cette partie.

Cas de la préemption Toute forme de préemption a été exclue de notre cadre d'étude. Néanmoins, la possibilité de préemption est présente dans de nombreuses études théoriques et applications industrielles des problèmes d'ordonnancement, et particulièrement en présence de maintenances. Dans le cadre de notre problème, différentes formes de préemption pourraient être considérées :

- Préemption des tâches de production : les tâches de production peuvent être interrompues pour effectuer d'autres tâches de production et reprendre plus tard.
- Préemption de la production par la maintenance : les tâches de production peuvent être interrompues pour effectuer une maintenance.

Dans ce cas, plusieurs hypothèses sont possibles pour définir l'impact de la préemption d'une tâche sur l'usure causée sur l'équipement : usure totale dès le début de l'exécution, usure proportionnelle à la durée passée sur la tâche, usure lorsque la tâche est finie, pénalisation

de la préemption... Il pourrait être intéressant de positionner ces différentes variantes dans la cartographie de la complexité proposée, et d'étudier dans quelle mesure les méthodes et modèles proposées peuvent s'appliquer à ces cas particuliers.

Approche de programmation par contraintes La programmation par contraintes est connue pour être particulièrement efficace pour la résolution de certains types de problèmes d'ordonnancement avec un objectif de type max (par exemple minimisation du temps de complétion du planning ou du retard maximum). En particulier, la programmation par contraintes dispose d'outils de modélisation très pratique pour les contraintes de ressources, qui peuvent être utilisés pour modéliser l'évolution de la santé de l'équipement. Néanmoins, une des particularités du problème est que lorsque la santé devrait régénérer plus que le maximum de santé, elle est automatiquement écrêtée, ce qui pose des difficultés pour la modélisation avec les outils cités précédemment. Il faudrait donc étudier si d'autres modélisations sont envisageables pour la santé, ou alors voir s'il est possible d'adapter les algorithmes de filtrage pour prendre en compte cette particularité.

Vers un PTAS pour la minimisation du délai moyen Nous avons vu que dans le cas où il est possible de regrouper les tâches en un ensemble de famille restreint de tâches de même usure, il est possible de résoudre le problème de minimisation du délai moyen en temps pseudo-polynomial ou polynomial par programmation dynamique. Cette propriété pourrait être exploitée dans un schéma d'approximation en temps polynomial (PTAS). Une idée similaire à l'idée utilisée pour le FPTAS du problème de sac à dos pourrait être utilisée ici. On notera toutefois qu'ici, le problème est NP-complet au sens fort, et qu'on l'on pourra au mieux obtenir un PTAS.

L'idée consiste à construire une suite d'heuristiques de la manière suivante. Pour la p-ième heuristique, on définit p familles de tâche de même usure. Pour chaque tâche, on arrondit au supérieur l'usure afin qu'elle puisse appartenir à une des familles. On obtient alors une suite d'heuristiques pseudo-polynomiales voire polynomiales, qui convergent finalement vers une méthode exacte (mais exponentielle) lorsqu'il y a autant de familles que de tâches. Afin d'obtenir un PTAS, il faudrait alors exprimer la complexité en fonction du ratio d'approximation des heuristiques. Cette étape est particulièrement difficile, puisqu'elle nécessite d'exprimer un ratio d'approximation pour les heuristiques, et donc de mesurer l'impact sur le délai moyen d'augmenter l'usure causée par certaines tâches.

Amélioration du modèle arc-flot Dans le cadre de l'étude de la minimisation du délai maximum avec seuils, nous avons adapté un modèle arc-flot pour la résolution du bin-packing à notre problème. Plusieurs améliorations peuvent être faites pour améliorer ce modèle. La taille du graphe sous-jacent peut-être réduite de manière significative en élaguant les sommets qui ne peuvent pas être atteints. Il est sans doute possible également d'appliquer la transformation proposée par Delorme et Iori (2020) pour réduire davantage la taille du graphe. La formulation arc-flot alternative, proposée par Brandão et Pedroso (2016), ne semble pas pouvoir être appliquée à notre problème, car elle nécessite une brisure de symétrie sur l'ordre des items, qui est incompatible avec notre contrainte de seuil.

Extensions de la génération de colonnes Nous avons également proposé une adaptation d'un algorithme de génération de colonnes de la littérature du bin-packing. Cette méthode heuristique nous permet d'obtenir rapidement de très bonnes solutions. Elle reste néanmoins améliorable sur plusieurs aspects.

La principale limitation actuellement est la reconstruction d'une solution entière par l'heuristique du problème maître à la fin de la procédure de génération de colonnes, qui en dépit de sa simplicité d'implémentation, représente la quasi-totalité du temps de calcul. Cette heuristique pourrait être remplacée par une heuristique plus sophistiquée, de type heuristique de diving. Une fois que l'optimum fractionnaire a été trouvé, les variables ayant des valeurs entières sont fixées, et la valeur non-entière la plus proche d'un entier est arrondie et fixée également. On relance ensuite la génération de colonnes et on réitère jusqu'à ce qu'une solution entière soit obtenue. Cette méthode permet d'éviter la résolution d'un programme linéaire en nombres entiers, et par conséquent est significativement plus rapide. De plus, la génération de colonnes est relancée après chaque arrondi, ce qui permet de générer des colonnes plus adaptées au fur et à mesure de la construction de la solution partielle, et donc d'obtenir de meilleures solutions.

Enfin, pour la résolution exacte du problème, la mise au point d'une méthode de type branch-and-price est une perspective naturelle de la génération de colonnes. Cette méthode repose sur le même principe que l'algorithme de diving, mais on effectue un branchement au lieu de faire un simple arrondi sur les variables non-entières. Cette méthode est connue pour être particulièrement efficace pour la résolution des problèmes de bin-packing.

Chapitre 6

Conclusion générale

Sommaire

6.1 Résumé des contributions

Dans cette thèse, nous avons abordé la thématique de l'optimisation des politiques de maintenance prédictive dans les systèmes de production à travers le prisme de la recherche opérationnelle, de l'optimisation mathématique et de l'algorithmique. Nous nous sommes plus particulièrement intéressés à la prise en compte d'un indicateur de santé des composants d'un équipement dans la modélisation et la résolution des problèmes d'aide à la décision considérés.

Dans la partie I, nous nous sommes intéressés au cas où la temporalité de la dégradation de l'équipement correspond à du moyen terme, de l'ordre du mois ou de l'année. Dans ce contexte, la question de la planification de la maintenance se pose au niveau tactique de décision. En établissant que la maintenance agit comme une forme de production différée, en prolongeant la durée d'utilisation de la machine sur le plus long terme, nous montrons que le problème de planification au niveau tactique de la maintenance partage de nombreuses similarités avec les problèmes de lot-sizing, habituellement rencontrés en planification de la production. Le problème étudié fait également intervenir un objectif de consommation de ressources original, incitant à garder un niveau de santé élevé tout au long du planning. Nous menons dans un premier temps une étude sur la complexité et la modélisation du problème : nous montrons que le problème est NP-difficile dans le cas général et proposons un modèle de programmation linéaire en nombres entiers bi-critère, partageant de nombreuses caractéristiques avec les modèles de lot-sizing utilisés en planification de la production.

Dans un second temps, nous nous intéressons à la résolution du problème bi-critère, pour lequel deux approches ont été considérées. La première approche consiste à demander au preneur de décision un budget maximum sur l'objectif de consommation de ressources, qui est alors traité comme une contrainte du problème. Le problème se ramène alors à la résolution du programme linéaire en nombres entiers avec le seul critère économique et la contrainte sur la consommation de ressources, qui peut être effectuée à l'aide d'un solveur. Nous montrons à l'aide d'expérimentations sur des données générées à partir de caractéristiques réalistes que cette approche permet de fournir des solutions avec une bonne garantie d'optimalité tant que le nombre de composants de l'équipement n'est pas trop important. La deuxième approche, quant à elle, consiste à proposer plusieurs solutions pertinentes du front de Pareto, représentatives du compromis entre les deux objectifs, au preneur de décision, qui effectue le choix a posteriori. Pour y parvenir, le front de Pareto est approximé par la méthode ε-contrainte, ce qui nécessite résoudre un grand nombre de programmes linéaires en nombres entiers. L'étude de la distribution des temps de résolution des points du front de Pareto a permis de mettre au point une méthodologie d'aide à la décision permettant de fournir efficacement un ensemble de diversifié de solutions avec une bonne garantie.

Dans la partie II, nous nous intéressons au cas où la temporalité de la dégradation de l'équipement est de l'ordre du jour ou de la semaine, ce qui place la prise de décision de la maintenance au niveau opérationnel. Nous nous concentrons alors sur un problème d'ordonnancement conjoint de tâches de production et de maintenance basé sur l'indicateur de santé de l'équipement. Nous proposons un système de notation et de classification unifié pour la prise en compte de la dégradation et de la maintenance dans les problèmes d'ordonnancement, basé sur les notations de Graham et les notations existantes pour les problèmes d'ordonnancement avec indisponibilité. Ce système nous permet de représenter efficacement les différents problèmes de la littérature, ainsi que le problème étudié et ses différentes variantes. Nous

établissons que les problèmes d'existence d'une solution, de minimisation du délai maximum et du délai moyen sont NP-difficiles au sens fort sur des cas minimaux peu restrictifs. Nous proposons une formulation de programmation linéaire en nombres entiers indexée par le temps pour la modélisation du problème et de ses différentes variantes. Nous étudions ensuite plus en détail les problèmes de minimisation du délai maximum et du délai moyen. Dans le cas du délai maximum, nous montrons que sous certaines hypothèses le problème est équivalent au problème de bin-packing classique, et notamment que les résultats concernant l'approximabilité du bin-packing sont conservés pour ce problème. Nous établissons également que dans le cas multi-composant, le ratio de compétitivité de tout algorithme online admet une borne inférieure dépendante du nombre de composants. Dans le cas de la minimisation du délai moyen, nous établissons différentes propriétés structurelles sur les solutions optimales du problèmes. Nous nous intéressons alors au cas où les tâches peuvent être regroupées dans un nombre limité de familles de tâches d'usure identique. En exploitant les propriétés du problème, nous établissons deux programmes dynamiques permettant de résoudre respectivement en temps pseudo-polynomial et polynomial cette variante, pour un nombre quelconque de familles de tâches.

Nous étudions ensuite plus spécifiquement la minimisation du délai maximum avec une contrainte de seuils d'acceptabilité sur la santé pour l'exécution des tâches. Ce problème est reformulé comme une généralisation du problème de bin-packing classique. Trois techniques de modélisation et de résolution du bin-packing sont adaptées pour prendre en compte la contrainte de seuils : la formulation polynomiale de Kantorovich, la génération de colonne basée sur la formulation "set-partitioning" et une formulation arc-flot. Les trois méthodes sont testées et comparées sur des jeux d'instances adaptés des benchmarks de la littérature. Il apparaît que la génération de colonnes et la formulation arc-flot sont très performantes sur les instances considérées.

En conclusion de chacune des deux parties de ce manuscrit, nous avons présenté des perspectives détaillées pour les deux problèmes que nous avons traités. Nous revenons à présent sur les principales perspectives concernant ces deux problèmes et présentons ensuite des perspectives à plus long terme concernant la thématique de cette thèse dans son ensemble.

6.2 Perspectives pour le niveau tactique de décision

6.2.1 Méthodes de résolution

Le problème de planification étudié dans la partie I est modélisé par programmation linéaire en nombres entiers et résolu à l'aide d'un solveur. Cette méthodologie permet d'obtenir des solutions optimales ou de bonnes solutions approchées pour des instances de taille limitée, mais passe difficilement à l'échelle pour des instances présentant un grand nombre de composants et un échantillonnage du temps important. Cette difficulté est amplifiée pour le calcul de front de Pareto dans le cas bi-critère, qui nécessite de résoudre le modèle de manière répétée.

L'une des principales perspectives de recherche est donc le développement d'algorithmes permettant d'obtenir rapidement de bonnes solutions. Des méthodes heuristiques, comme des approches gloutonnes combinées à de la recherche locale, sont une piste prometteuse et pourraient également fournir des bornes pour améliorer la résolution par programmation

linéaire en nombres entiers. La piste de méthodes de résolution exactes dédiées de type branch-and-bound demandent un effort de développement plus important, mais pourraient également permettre une amélioration de la résolution.

6.2.2 Méthodologie de décision bi-crière

Outre la difficulté liée à la résolution du modèle ε-contraint, de nombreux défis restent à relever pour parvenir à un outil d'aide à la décision efficace pour le problème bi-critère. Une des principales pistes d'amélioration porte sur l'échantillonnage du front de Pareto par la méthode ε-contrainte. La progression arithmétique proposée entraine le sous-échantillonnage d'une partie du front et le sur-échantillonnage de l'autre. L'étude de pas d'échantillonnage variables, comme, par exemple, une progression géométrique, pourrait aboutir à des ensembles de solutions plus représentatifs du compromis. Enfin, nous avons vu que la méthode algorithmique de sélections de points sur le front de Pareto était peu satisfaisante par rapport à la méthode visuelle. Cette méthode pourrait donc être améliorée en intégrant une mesure d'homogénéité pour obtenir des résultats se rapprochant de ceux obtenus par visualisation du front.

6.3 Perspectives pour le niveau opérationnel de décision

6.3.1 Problème d'ordonnancement

Nous avons introduit dans la partie II un problème générique d'ordonnancement conjoint de la production et de la maintenance en présence d'un indicateur de santé. Nous nous sommes ensuite concentrés plus spécifiquement sur quelques variantes de ce problème, et un travail important reste à faire sur l'étude des différents objectifs et contraintes qui n'ont pas été traités dans cette thèse : on peut notamment penser au cas des problèmes à machines parallèles ou aux problèmes d'ordonnancement d'atelier, à des contraintes classiques telles que les dates de libération et de livraison, et à des objectifs comme le retard maximum.

Concernant la minimisation du délai maximum, une approche de programmation par contrainte pourrait s'avérer particulièrement efficace en modélisant les contraintes de santé des équipements comme des contraintes de ressources. Toutefois, le phénomène d'écrêtage lorsque la santé est régénérée au-dessus du maximum pose des difficultés de modélisation et la conception de nouveaux algorithmes de filtrages pour sa prise en compte pourrait être nécessaire pour y parvenir.

Du côté de la minimisation du délai moyen, les différentes propriétés structurelles ont permis de mettre au point deux schémas de programmation dynamique. Ces schémas de programmation dynamique pourraient être intégrés dans un PTAS, où les tâches dont l'usure est proche seraient regroupées dans des familles d'usure identique avec arrondi au supérieur. Certaines difficultés subsistent toutefois sur la manière de définir les familles, et sur la mise en équation de l'impact de ce regroupement sur la valeur de l'objectif. Les propriétés du problème pourraient également être exploitées par exemple dans un algorithme de branch-and-bound dédié, ou pour des heuristiques.

6.3.2 Problème de Bin-Packing

Enfin, du côté du problème de Bin-Packing avec Seuils, cas particulier de notre problème d'ordonnancement générique, plusieurs perspectives de recherche peuvent également être envisagées. La formulation arc-flot a montré qu'elle était très efficace dans toutes les situations considérées tant que la capacité des boites n'est pas trop importante. Néanmoins, lorsque la capacité est élevée, le nombre de variables devient trop important pour permettre la résolution du modèle par un solveur. Cette difficulté peut être surmontée en observant que le graphe sous-jascent peut contenir de nombreux sommets inatteignables, qui pourraient être élagués afin de réduire le nombre de variables. Les différentes techniques récentes investiguées dans le cas du Bin-Packing classique pourraient également être étudiées.

La méthode de génération de colonnes s'est également montrée très performante. De nombreuses extensions sont possibles, en particulier, l'implémentation d'algorithmes de diving pour la reconstruction d'une solution entière ou l'intégration à un algorithme de branch-and-price pour la résolution exacte sont à considérer.

6.4 Perspectives générales de recherche

6.4.1 Vers une vision plus globale

Les deux problèmes que nous avons étudiés dans le cadre de cette thèse sont des sous-problèmes d'un problème plus large de planification de la maintenance, présenté dans le premier chapitre. Si les deux sous-problèmes sont liés, ils traitent de systèmes dont la temporalité de la dégradation est différente : de l'ordre de la semaine ou du mois pour le problème étudié dans la première partie, de l'ordre du jour pour celui de la deuxième partie.

À plus long terme, il serait intéressant d'aller vers une approche intégrée permettant de résoudre le problème dans son ensemble pour un cas d'étude industriel, par exemple dans le milieu des semi-conducteurs. Cela passe par l'étude du niveau stratégique de décision, et suivant la temporalité de la dégradation, définir les sous-problèmes correspondants, et de les intégrer dans une méthodologie d'aide à la décision globale.

6.4.2 Prise en compte de l'incertitude

L'optimisation de la maintenance dans les systèmes de production est un sujet particulièrement complexe en pratique notamment en raison de la nature stochastique inhérente de certains aspects du problème : la dégradation et les pannes sont des phénomènes qui ne sont pas toujours prévisibles et leur prise en compte est un obstacle majeur pour l'optimisation des politiques de maintenance. Dans cette thèse, nous avons supposé que tout le travail de gestion de l'incertitude avait été fait en amont au niveau stratégique dans la méthodologie de calcul de l'indicateur de santé, par exemple à partir des données constructeur des équipements, et n'avons considéré que des problèmes déterministes. Si le paradigme de la maintenance prédictive apporte une meilleure maîtrise de l'évolution de l'état de l'équipement au cours du temps, la supposition que la dégradation est parfaitement contrôlée reste une hypothèse forte. La problématique de la réoptimisation des plannings suite à un événement imprévu pourrait être une piste de recherche particulièrement intéressante pour mieux prendre en

compte cette incertitude. Enfin, des approches robustes basées sur des scénarios sont une approche différente de la prise en compte de l'incertitude qui peuvent s'avérer particulièrement intéressantes dans des contextes où les pannes peuvent avoir des conséquences particulièrement importantes. Ces approches sont d'autant plus intéressantes dans le cadre de stratégies de maintenance préventives ou prédictives, où l'on dispose de beaucoup de connaissances sur le fonctionnement de l'équipement, ce qui facilite l'élaboration des scénarios.

Bibliographie

Absi, N., Dauzère-Pérès, S., Kedad-Sidhoum, S., Penz, B., & Rapine, C. (2013). Lot sizing with carbon emission constraints. *European Journal of Operational Research*, *227*(1), 55-61. https://doi.org/10.1016/j.ejor.2012.11.044

Adiri, I., Bruno, J., Frostig, E., & Rinnooy Kan, A. H. G. (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, *26*(7), 679-696. https://doi.org/10.1007/BF00288977

Ahmad, R., & Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & industrial engineering*, *63*(1), 135-149. https://doi.org/10.1016/j.cie.2012.02.002

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1988). Network flows. https://doi.org/10.5555/137406

Akbalik, A., Penz, B., & Rapine, C. (2015a). Capacitated lot sizing problems with inventory bounds. *Annals of Operations Research*, *229*, 1-18. https://doi.org/10.1007/s10479-015-1816-6

Akbalik, A., Penz, B., & Rapine, C. (2015b). Multi-item uncapacitated lot sizing problem with inventory bounds. *Optimization Letters*, *9*, 143-154. https://doi.org/10.1007/s11590-014-0746-6

Akbalik, A., & Rapine, C. (2012). Polynomial time algorithms for the constant capacitated single-item lot sizing problem with stepwise production cost. *Operations Research Letters*, *40*(5), 390-397. https://doi.org/doi.org/10.1016/j.orl.2012.05.003

Akbalik, A., & Rapine, C. (2013). The single item uncapacitated lot-sizing problem with time-dependent batch sizes : NP-hard and polynomial cases. *European Journal of Operational Research*, *229*(2), 353-363. https://doi.org/10.1016/j.ejor.2013.02.052

Akturk, M. S., Ghosh, J. B., & Gunes, E. D. (2003). Scheduling with tool changes to minimize total completion time : A study of heuristics and their performance. *Naval Research Logistics (NRL)*, *50*(1), 15-30. https://doi.org/10.1002/nav.10045

Akturk, M., Ghosh, J. B., & Gunes, E. D. (2004). Scheduling with tool changes to minimize total completion time : Basic results and SPT performance. *European Journal of Operational Research*, *157*(3), 784-790. https://doi.org/10.1016/S0377-2217(03)00232-7

Al-Turki, U. M., Ayar, T., Yilbas, B. S., & Sahin, A. Z. (2014). *Integrated maintenance planning*. Springer. https://doi.org/10.1007/978-3-319-06290-7

Atamtürk, A., & Küçükyavuz, S. (2008). An O (n2) algorithm for lot sizing with inventory bounds and fixed costs. *Operations Research Letters*, *36*(3), 297-299. https://doi.org/10.1016/j.orl.2007.08.004

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European journal of operational research*, *292*(2), 397-422. https://doi.org/10.1016/j.ejor.2020.11.016

Azadnia, A. H., Saman, M. Z. M., & Wong, K. Y. (2015). Sustainable supplier selection and order lot-sizing : an integrated multi-objective decision-making process. *International Journal of Production Research*, *53*(2), 383-408. https://doi.org/10.1080/00207543.2014.935827

Baldi, M. M., Heinicke, F., Simroth, A., & Tadei, R. (2016). New heuristics for the stochastic tactical railway maintenance problem. *Omega*, *63*, 94-102. https://doi.org/10.1016/j.omega.2015.10.005

Balogh, J., Epstein, L., & Levin, A. (2021). More on ordered open end bin packing. *Journal of Scheduling*, *24*(6), 589-614. https://doi.org/10.1007/s10951-021-00709-3

Barany, I., Van Roy, T. J., & Wolsey, L. A. (1984). Strong formulations for multi-item capacitated lot sizing. *Management Science*, *30*(10), 1255-1261. https://doi.org/10.1287/mnsc.30.10.1255

Baur, M., Albertelli, P., & Monno, M. (2020). A review of prognostics and health management of machine tools. *The International Journal of Advanced Manufacturing Technology*, *107*, 2843-2863. https://doi.org/10.1007/s00170-020-05202-3

Belov, G., & Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research*, *171*(1), 85-106. https://doi.org/https://doi.org/10.1016/j.ejor.2004.08.036

Beume, N., Fonseca, C. M., Lopez-Ibanez, M., Paquete, L., & Vahrenhold, J. (2009). On the Complexity of Computing the Hypervolume Indicator. *IEEE Transactions on Evolutionary Computation*, *13*(5), 1075-1082. https://doi.org/10.1109/TEVC.2009.2015575

Brahimi, N., Absi, N., Dauzère-Pérès, S., & Nordli, A. (2017). Single-item dynamic lot-sizing problems : An updated survey. *European Journal of Operational Research*, *263*(3), 838-863. https://doi.org/10.1016/j.ejor.2017.05.008

Brahimi, N., Dauzere-Peres, S., Najid, N. M., & Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research*, *168*(1), 1-16. https://doi.org/10.1016/j.ejor.2004.01.054

Brandão, F., & Pedroso, J. P. (2016). Bin packing and related problems : General arc-flow formulation with graph compression. *Computers & Operations Research*, *69*, 56-67. https://doi.org/10.1016/j.cor.2015.11.009

Breit, J., Schmidt, G., & Strusevich, V. A. (2003). Non-preemptive two-machine open shop scheduling with non-availability constraints. *Mathematical Methods of Operations Research (ZOR)*, *57*(2), 217-234. https://doi.org/10.1007/s001860200267

Bridonneau, G. (2022). *Joint Production/Maintenance scheduling with consideration of the Health Index* (mém. de mast.). Université Grenoble-Alpes.

Cambazard, H., & O'Sullivan, B. (2010). Propagating the Bin Packing Constraint Using Linear Programming. In D. Cohen (Éd.), *Principles and Practice of Constraint Programming – CP 2010* (p. 129-136). Springer. https://doi.org/10.1007/978-3-642-15396-9_13

Cassady, C. R., & Kutanoglu, E. (2003). Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. *IIE transactions*, *35*(6), 503-513. https://doi.org/10.1080/07408170304416

Chen, A., & Wu, G. S. (2007). Real-time health prognosis and dynamic preventive maintenance policy for equipment under aging Markovian deterioration. *International Journal of Production Research*, *45*(15), 3351-3379. https://doi.org/10.1080/00207540600677617

Chen, W. J. (2006). Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society*, *57*(4), 410-415. https://doi.org/10.1057/palgrave.jors.2601998

Clautiaux, F., Sadykov, R., Vanderbeck, F., & Viaud, Q. (2019). Pattern-based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. *EURO Journal on Computational Optimization*, *7*(3), 265-297. https://doi.org/https://doi.org/10.1007/s13675-019-00113-9

Coffman, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms : Survey and classification. In *Handbook of Combinatorial Optimization* (p. 455-531). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_35

Cook, S. A. (1971). The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, 143-152. https://doi.org/10.1145/3588287.3588297

D'Ariano, A., Meng, L., Centulio, G., & Corman, F. (2019). Integrated stochastic optimization approaches for tactical scheduling of trains and railway infrastructure maintenance. *Computers & Industrial Engineering*, *127*, 1315-1335. https://doi.org/10.1016/j.cie.2017.12.010

De Lima, V. L., Alves, C., Clautiaux, F., Iori, M., & Valério de Carvalho, J. M. (2022). Arc flow formulations based on dynamic programming : Theoretical foundations and applications. *European Journal of Operational Research*, *296*(1), 3-21. https://doi.org/10.1016/j.ejor.2021.04.024

De Lima, V. L., Iori, M., & Miyazawa, F. K. (2023). Exact solution of network flow models with strong relaxations. *Mathematical Programming*, *197*(2), 813-846. https://doi.org/10.1007/s10107-022-01785-9

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805-824. https://doi.org/10.1016/j.ejor.2019.09.047

Delorme, M., & Iori, M. (2020). Enhanced Pseudo-polynomial Formulations for Bin Packing and Cutting Stock Problems. *INFORMS Journal on Computing*, *32*(1), 101-119. https://doi.org/10.1287/ijoc.2018.0880

Delorme, M., Iori, M., & Martello, S. (2016). Bin packing and cutting stock problems : Mathematical models and exact algorithms. *European Journal of Operational Research*, *255*(1), 1-20. https://doi.org/10.1016/j.ejor.2016.04.030

de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety*, *221*, 108341. https://doi.org/10.1016/j.ress.2022.108341

Díaz-Madroñero, M., Mula, J., & Peidro, D. (2014). A review of discrete-time optimization models for tactical production planning. *International Journal of Production Research*, *52*(17), 5171-5205. https://doi.org/10.1080/00207543.2014.899721

Dyckhoff, H. (1981). A New Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, *29*(6), 1092-1104. https://doi.org/10.1287/opre.29.6.1092

Fitouhi, M.-C., & Nourelfath, M. (2014). Integrating noncyclical preventive maintenance scheduling and production planning for multi-state systems. *Reliability Engineering & System Safety*, *121*, 175-186. https://doi.org/10.1016/j.ress.2013.07.009

Fonseca, C., Paquete, L., & Lopez-Ibanez, M. (2006). An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator. *2006 IEEE International Conference on Evolutionary Computation*, 1157-1163. https://doi.org/10.1109/CEC.2006.1688440

Foussard, E. (2021). *Maintenance Planning for Circular Economy : Laundromat Washing Machines Case* (Technical Report). G-SCOP - Laboratoire des sciences pour la conception, l'optimisation et la production. https://hal.science/hal-03151214

Foussard, E., Espinouse, M.-L., Mounié, G., & Nattaf, M. (2021). A lot-sizing model for maintenance planning in a circular economy context. *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems : IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part II*, 673-682. https://doi.org/10.1007/978-3-030-85902-2_72

Froger, A., Gendreau, M., Mendoza, J. E., Pinson, É., & Rousseau, L.-M. (2016). Maintenance scheduling in the electricity industry : A literature review. *European Journal of Operational Research*, *251*(3), 695-706. https://doi.org/10.1016/j.ejor.2015.08.045

Gao, K., Peng, R., Qu, L., & Wu, S. (2020). Jointly optimizing lot sizing and maintenance policy for a production system with two failure modes. *Reliability Engineering & System Safety*, *202*, 106996. https://doi.org/10.1016/j.ress.2020.106996

Garey, M. R., Graham, R. L., Johnson, D. S., & Yao, A. C.-C. (1976). Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory, Series A*, *21*(3), 257-298. https://doi.org/10.1016/0097-3165(76)90001-7

Garey, M. R., & Johnson, D. S. (1990). *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. https://doi.org/10.5555/574848

Geng, S., & Wang, X. (2022). Predictive maintenance scheduling for multiple power equipment based on data-driven fault prediction. *Computers & Industrial Engineering*, *164*, 107898. https://doi.org/10.1016/j.cie.2021.107898

Gerum, P. C. L., Altay, A., & Baykal-Gürsoy, M. (2019). Data-driven predictive maintenance scheduling policies for railways. *Transportation Research Part C : Emerging Technologies*, *107*, 137-154. https://doi.org/10.1016/j.trc.2019.07.020

Giglio, D., Paolucci, M., & Roshani, A. (2017). Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems. *Journal of cleaner production*, *148*, 624-641. https://doi.org/10.1016/j.jclepro.2017.01.166

Gilmore, P. C., & Gomory, R. E. (1961). A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, *9*(6), 849-859. https://doi.org/10.1287/opre.9.6.849

Glock, C. H., Grosse, E. H., & Ries, J. M. (2014). The lot sizing problem : A tertiary study. *International Journal of Production Economics*, *155*, 39-51. https://doi.org/10.1016/j.ijpe.2013.12.009

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling : a Survey. In P. Hammer, E. Johnson & B. Korte (Éd.), *Discrete Optimization II* (p. 287-326). Elsevier. https://doi.org/https://doi.org/10.1016/S0167-5060(08)70356-X

Grall, A., Dieulle, L., Bérenguer, C., & Roussignol, M. (2002). Continuous-time predictive-maintenance scheduling for a deteriorating system. *IEEE transactions on reliability*, *51*(2), 141-150. https://doi.org/10.1109/TR.2002.1011518

Hajej, Z., & Rezg, N. (2020). An optimal integrated lot sizing and maintenance strategy for multi-machines system with energy consumption. *International Journal of Production Research*, *58*(14), 4450-4470. https://doi.org/10.1080/00207543.2019.1654630

Hassankhani Dolatabadi, S., & Budinska, I. (2021). Systematic literature review predictive maintenance solutions for SMEs from the last decade. *Machines*, *9*(9), 191. https://doi.org/10.3390/machines9090191

He, P., Zhang, W., Xu, X., & Bian, Y. (2015). Production lot-sizing and carbon emissions under cap-and-trade and carbon tax regulations. *Journal of Cleaner Production*, *103*, 241-248. https://doi.org/10.1016/j.jclepro.2014.08.102

He, Y., Zhong, W., & Gu, H. (2006). Improved algorithms for two single machine scheduling problems. *Theoretical Computer Science*, *363*(3), 257-265. https://doi.org/10.1016/j.tcs.2006.04.014

Hwang, H.-C., & Van Den Heuvel, W. (2012). Improved algorithms for a lot-sizing problem with inventory bounds and backlogging. *Naval Research Logistics (NRL)*, *59*(3-4), 244-253. https://doi.org/10.1002/nav.21485

Ishibuchi, H., Imada, R., Setoguchi, Y., & Nojima, Y. (2018). How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison. *Evolutionary Computation*, *26*(3), 411-440. https://doi.org/10.1162/evco_a_00226

Jafari, L., & Makis, V. (2015). Joint optimal lot sizing and preventive maintenance policy for a production facility subject to condition monitoring. *International Journal of Production Economics*, *169*, 156-168. https://doi.org/10.1016/j.ijpe.2015.07.034

Jans, R., & Degraeve, Z. (2008). Modeling industrial lot sizing problems : a review. *International Journal of Production Research*, *46*(6), 1619-1643. https://doi.org/10.1080/00207540600902262

Ji, M., He, Y., & Cheng, T. C. E. (2007). Single-machine scheduling with periodic maintenance to minimize makespan. *Computers & Operations Research*, *34*(6), 1764-1770. https://doi.org/10.1016/j.cor.2005.05.034

Jing, F., & Mu, Y. (2020). Dynamic lot-sizing model under perishability, substitution, and limited storage capacity. *Computers & Operations Research*, *122*, 104978. https://doi.org/10.1016/j.cor.2020.104978

Kaabi, J., & Harrath, Y. (2014). A Survey of Parallel Machine Scheduling under Availability Constraints. *International Journal of Computer and Information Technology*, *03*(02), 8.

Kacem, I. (2009). Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. *Journal of Combinatorial Optimization*, *17*(2), 117-133. https://doi.org/10.1007/s10878-007-9102-4

Kacem, I., & Haouari, M. (2008). Approximation algorithms for single machine scheduling with one unavailability period. *4OR*, *7*(1), 79. https://doi.org/10.1007/s10288-008-0076-6

Kacem, I., & Mahjoub, A. R. (2009). Fully polynomial time approximation scheme for the weighted flow-time minimization on a single machine with a fixed non-availability interval. *Computers & Industrial Engineering*, *56*(4), 1708-1712. https://doi.org/10.1016/j.cie.2008.09.042

Kantorovich, L. V. (1960). Mathematical Methods of Organizing and Planning Production. *Management Science*, *6*(4), 366-422. https://doi.org/10.1287/mnsc.6.4.366

Kao, Y.-T., Dauzère-Pérès, S., Blue, J., & Chang, S.-C. (2018). Impact of integrating equipment health in production scheduling for semiconductor fabrication. *Computers & Industrial Engineering*, *120*, 450-459. https://doi.org/10.1016/j.cie.2018.04.053

Karimi, B., Fatemi Ghomi, S., & Wilson, J. (2003). The capacitated lot sizing problem : a review of models and algorithms. *Omega*, *31*(5), 365-378. https://doi.org/10.1016/S0305-0483(03)00059-8

Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, *56*(1), 357-367. https://doi.org/https://doi.org/10.1016/j.cie.2008.06.008

Keysan, G., Nemhauser, G. L., & Savelsbergh, M. W. (2010). Tactical and operational planning of scheduled maintenance for per-seat, on-demand air transportation. *Transportation Science*, *44*(3), 291-306. https://doi.org/10.1287/trsc.1090.0311

Lageweg, B., Lenstra, J. K., Lawler, E., & Kan, A. R. (1982). Computer-Aided complexity classification of combinational problems. *Communications of the ACM*, *25*(11), 817-822. https://doi.org/10.1145/358690.363066

Lee, C.-Y. (1996). Machine scheduling with an availability constraint. *Journal of global optimization*, *9*(3-4), 395-416. https://doi.org/10.1007/BF00121681
Papier fondateur. Tables de complexité pour les différents problèmes.

Lee, C.-Y., & Liman, S. D. (1992). Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica*, *29*(4), 375-382. https://doi.org/10.1007/BF01178778

Lemoine, D., & Castanier, B. (2021). Tactical Planning and Predictive Maintenance : Towards an Integrated Model Based on ε-reliability. *IFIP International Conference on Advances in Production Management Systems*, 92-101. https://doi.org/10.1007/978-3-030-85874-2_10

Li, M., & Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation : A survey. *ACM Computing Surveys (CSUR)*, *52*(2), 1-38. https://doi.org/10.1145/3300148

Li, Y., Peng, S., Li, Y., & Jiang, W. (2020). A review of condition-based maintenance : Its prognostic and operational aspects. *Frontiers of Engineering Management*, *7*(3), 323-334. https://doi.org/10.1007/s42524-020-0121-5

Lippman, S. A. (1969). Optimal inventory policy with multiple set-up costs. *Management Science*, *16*(1), 118-138. https://doi.org/10.1287/mnsc.16.1.118

Love, S. F. (1973). Bounded production and inventory models with piecewise concave costs. *Management Science*, *20*(3), 313-318. https://doi.org/10.1287/mnsc.20.3.313

Low, C., Ji, M., Hsu, C.-J., & Su, C.-T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, *34*(2), 334-342. https://doi.org/10.1016/j.apm.2009.04.014

Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199-211. https://doi.org/10.1016/j.cie.2009.04.014

Malik, M. A. K. (1979). Reliable preventive maintenance scheduling. *AIIE transactions*, *11*(3), 221-228. https://doi.org/10.1080/05695557908974463

Marquez, A. C., & Gupta, J. N. (2006). Contemporary maintenance management : process, framework and supporting pillars. *Omega*, *34*(3), 313-326. https://doi.org/10.1016/j.omega.2004.11.003

Martello, S., & Toth, P. (1990). *Knapsack Problems : Algorithms and Computer Implementations*. John Wiley & Sons, Inc. https://doi.org/10.5555/98124

Martinez, C. (2020). *Considération de la dimension humaine dans l'optimisation de tournées de soins et services à domicile soumises à des perturbations* (thèse de doct.) [2020GRALI060]. http://www.theses.fr/2020GRALI060/document

Masmoudi, M., & Haït, A. (2010). A tactical model under uncertainty for helicopter maintenance planning. *the 8th ENIM IFAC International Conference of Modeling and Simulation, MOSIM*, *10*, 1837-1845.

Masmoudi, O., Yalaoui, A., Ouazene, Y., & Chehade, H. (2017). Lot-sizing in a multi-stage flow line production system with energy consideration. *International Journal of Production Research*, *55*(6), 1640-1663. https://doi.org/10.1080/00207543.2016.1206670

Mintzberg, H., Raisinghani, D., & Theoret, A. (1976). The structure of" unstructured" decision processes. *Administrative science quarterly*, 246-275.

Moghaddam, K. S., & Usher, J. S. (2011). Preventive maintenance and replacement scheduling for repairable and maintainable systems using dynamic programming. *Computers & Industrial Engineering*, *60*(4), 654-665. https://doi.org/https://doi.org/10.1016/j.cie.2010.12.021

Murthy, D., Atrens, A., & Eccleston, J. (2002). Strategic maintenance management. *Journal of Quality in Maintenance Engineering*, *8*(4), 287-305. https://doi.org/10.1108/13552510210448504

Olde Keizer, M. C., Flapper, S. D. P., & Teunter, R. H. (2017). Condition-based maintenance policies for systems with multiple dependent components : A review. *European Journal of Operational Research*, *261*(2), 405-420. https://doi.org/10.1016/j.ejor.2017.02.044

Peng, H., & van Houtum, G.-J. (2016). Joint optimization of condition-based maintenance and production lot-sizing. *European Journal of Operational Research*, *253*(1), 94-107. https://doi.org/10.1016/j.ejor.2016.02.027

Penz, L., Dauzère-Pérès, S., & Nattaf, M. (2023). Minimizing the sum of completion times on a single machine with health index and flexible maintenance operations. *Computers & Operations Research*, *151*, 106092. https://doi.org/https://doi.org/10.1016/j.cor.2022.106092

Phouratsamay, S.-L., Kedad-Sidhoum, S., & Pascual, F. (2018). Two-level lot-sizing with inventory bounds. *Discrete Optimization*, *30*, 1-19. https://doi.org/10.1016/j.disopt.2018.05.001

Pinedo, M. L. (2012). *Scheduling* (T. 29). Springer. https://doi.org/10.1007/978-1-4614-2361-4

Pintelon, L. M., & Gelders, L. (1992). Maintenance management decision making. *European journal of operational research*, *58*(3), 301-317. https://doi.org/10.1016/0377-2217(92)90062-E

Prajapati, A., Bechtel, J., & Ganesan, S. (2012). Condition based maintenance : a survey. *Journal of Quality in Maintenance Engineering*, *18*(4), 384-400. https://doi.org/10.1108/13552511211281552

Qi, X., Chen, T., & Tu, F. (1999). Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, *50*(10), 1071-1078. https://doi.org/10.1057/palgrave.jors.2600791

Qi, X. (2007). A note on worst-case performance of heuristics for maintenance scheduling problems. *Discrete Applied Mathematics*, *155*(3), 416-422. https://doi.org/10.1016/j.dam.2006.06.005

Quatrini, E., Costantino, F., Di Gravio, G., & Patriarca, R. (2020). Condition-based maintenance—an extensive literature review. *Machines*, *8*(2), 31. https://doi.org/10.3390/machines8020031

Rao, M. R. (1976). On the cutting stock problem. *Journal of the Computer Society of India*, *7*, 35-39.

Rapine, C., Penz, B., Gicquel, C., & Akbalik, A. (2018). Capacity acquisition for the single-item lot sizing problem under energy constraints. *Omega*, *81*, 112-122. https://doi.org/10.1016/j.omega.2017.10.004

Retel Helmrich, M. J., Jans, R., van den Heuvel, W., & Wagelmans, A. P. (2015). The economic lot-sizing problem with an emission capacity constraint. *European Journal of Operational Research*, *241*(1), 50-62. https://doi.org/10.1016/j.ejor.2014.06.030

Roda, I., & Macchi, M. (2021). Maintenance concepts evolution : A comparative review towards advanced maintenance conceptualization. *Computers in Industry*, *133*, 103531. https://doi.org/10.1016/j.compind.2021.103531

Sadfi, C., Penz, B., Rapine, C., Błażewicz, J., & Formanowicz, P. (2005). An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. *European Journal of Operational Research*, *161*(1), 3-10. https://doi.org/10.1016/j.ejor.2003.08.026

Sadykov, R., & Vanderbeck, F. (2013). Bin Packing with Conflicts : A Generic Branch-and-Price Algorithm. *INFORMS Journal on Computing*, *25*(2), 244-255. https://doi.org/10.1287/ijoc.1120.0499

Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., & Uchoa, E. (2019). Primal Heuristics for Branch and Price : The Assets of Diving Methods. *INFORMS Journal on Computing*, *31*(2), 251-267. https://doi.org/10.1287/ijoc.2018.0822

Sanlaville, E., & Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*(9), 795-811. https://doi.org/10.1007/s002360050143

Scheithauer, G., & Terno, J. (1997). Theoretical investigations on the modified integer round-up property for the one-dimensional cutting stock problem. *Operations Research Letters*, *20*, 93-100. https://doi.org/10.1016/S0167-6377(96)00047-8

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1-15. https://doi.org/10.1016/S0377-2217(98)00367-1

Schoenfield, J. E. (2002). Fast, exact solution of open bin packing problems without linear programming. *Draft, US Army Space and Missile Defense Command, Huntsville, Alabama, USA*.

Scholl, A., Klein, R., & Jürgens, C. (1997). Bison : A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, *24*(7), 627-645. https://doi.org/https://doi.org/10.1016/S0305-0548(96)00082-2

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons. https://doi.org/10.5555/17634

Shamsaei, F., & Van Vyve, M. (2017). Solving integrated production and condition-based maintenance planning problems by MIP modeling. *Flexible Services and Manufacturing Journal*, *29*, 184-202. https://doi.org/10.1007/s10696-016-9244-8

Shang, K., Ishibuchi, H., He, L., & Pang, L. M. (2021). A Survey on the Hypervolume Indicator in Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, *25*(1), 1-20. https://doi.org/10.1109/TEVC.2020.3013290

Shaw, K. (2017). Fuzzy multi-objective, multi-item, multi-supplier, lot-sizing considering carbon footprint. *International Journal of Mathematics in Operational Research*, *11*(2), 171-203. https://doi.org/10.1504/IJMOR.2017.086289

Shaw, P. (2004). A Constraint for Bin Packing. In M. Wallace (Éd.), *Principles and Practice of Constraint Programming – CP 2004* (p. 648-662). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30201-8_47

Strusevich, V. A., Rustogi, K., et al. (2017). *Scheduling with time-changing effects and rate-modifying activities*. Springer. https://doi.org/10.1007/978-3-319-39574-6

Suzanne, E., Absi, N., & Borodin, V. (2020). Towards circular economy in production planning : Challenges and opportunities. *European Journal of Operational Research*, *287*(1), 168-190. https://doi.org/10.1016/j.ejor.2020.04.043

T'kindt, V., & Billaut, J.-C. (2006). Multicriteria optimisation theory. In *Multicriteria Scheduling : Theory, Models and Algorithms* (p. 53-112). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-24789-0_4

Tsang, A. H. (2002). Strategic dimensions of maintenance management. *Journal of Quality in maintenance Engineering*, *8*(1), 7-39. https://doi.org/10.1108/13552510210420577

Valério de Carvalho, J. M. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, *141*(2), 253-273. https://doi.org/10.1016/S0377-2217(02)00124-8

Valério de Carvalho, J. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, *86*, 629-659. https://doi.org/10.1023/A:1018952112615

Van Leeuwen, J. (1991). *Handbook of theoretical computer science (vol. A) algorithms and complexity*. Mit Press. https://doi.org/10.5555/114872

Van Veldhuizen, D. A., & Lamont, G. B. (2000). On measuring multiobjective evolutionary algorithm performance. *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, *1*, 204-211. https://doi.org/10.1109/CEC.2000.870296

Van Vyve, M. (2007). Algorithms for single-item lot-sizing problems with constant batch size. *Mathematics of Operations Research*, *32*(3), 594-613. https://doi.org/10.1287/moor.1070.0257

Vance, P. H. (1998). Branch-and-Price Algorithms for the One-Dimensional Cutting Stock Problem. *Computational Optimization and Applications*, *9*(3), 211-228. https://doi.org/10.1023/A:1018346107246

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, *5*(1), 89-96. https://doi.org/10.1287/mnsc.5.1.89

Wang, G., Sun, H., & Chu, C. (2005). Preemptive Scheduling with Availability Constraints to Minimize Total Weighted Completion Times. *Annals of Operations Research*, *133*(1-4), 183-192. https://doi.org/10.1007/s10479-004-5032-z

Wang, W. (2012). An overview of the recent advances in delay-time-based maintenance modelling. *Reliability Engineering & System Safety*, *106*, 165-178. https://doi.org/10.1016/j.ress.2012.04.004

Wee, T. S., & Magazine, M. J. (1982). Assembly line balancing as generalized bin packing. *Operations Research Letters*, *1*(2), 56-58. https://doi.org/10.1016/0167-6377(82)90046-3

Wichmann, M. G., Johannes, C., & Spengler, T. S. (2019). An extension of the general lot-sizing and scheduling problem (GLSP) with time-dependent energy prices. *Journal of Business Economics*, *89*, 481-514. https://doi.org/10.1007/s11573-018-0921-9

Witt, A. (2019). A heuristic for the multi-level capacitated lot sizing problem with inventory constraints. *International Journal of Management Science and Engineering Management*, *14*(4), 240-252. https://doi.org/10.1080/17509653.2018.1555495

Wu, Y., Maravelias, C. T., Wenzel, M. J., ElBsat, M. N., & Turney, R. T. (2021). Predictive maintenance scheduling optimization of building heating, ventilation, and air conditioning systems. *Energy and Buildings*, *231*, 110487. https://doi.org/10.1016/j.enbuild.2020.110487

Yang, L., Ye, Z.-s., Lee, C.-G., Yang, S.-f., & Peng, R. (2019). A two-phase preventive maintenance policy considering imperfect repair and postponed replacement. *European Journal of Operational Research*, *274*(3), 966-977. https://doi.org/https://doi.org/10.1016/j.ejor.2018.10.049

Yao, X., Fernández-Gaucherand, E., Fu, M. C., & Marcus, S. I. (2004). Optimal preventive maintenance scheduling in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, *17*(3), 345-356. https://doi.org/10.1109/TSM.2004.831948

Yazdani, M., Khalili, S. M., Babagolzadeh, M., & Jolai, F. (2017). A single-machine scheduling problem with multiple unavailability constraints : A mathematical model and an enhanced variable neighborhood search approach. *Journal of Computational Design and Engineering*, *4*(1), 46-59. https://doi.org/10.1016/j.jcde.2016.08.001

Yuan, J., Shi, L., & Ou, J. (2008). Single machine scheduling with forbidden intervals and job delivery times [Publisher : World Scientific Publishing Co.]. *Asia-Pacific Journal of Operational Research*, *25*(03), 317-325. https://doi.org/10.1142/S0217595908001778

Zhao, J., Gao, C., & Tang, T. (2022). A Review of Sustainable Maintenance Strategies for Single Component and Multicomponent Equipment. *Sustainability*, *14*(5). https://doi.org/10.3390/su14052992

Zhou, X., Xi, L., & Lee, J. (2007). Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation. *Reliability engineering & system safety*, *92*(4), 530-534. https://doi.org/10.1016/j.ress.2006.01.006

Zio, E., & Compare, M. (2013). Evaluating maintenance policies by quantitative modeling and analysis. *Reliability Engineering & System Safety*, *109*, 53-65. https://doi.org/10.1016/j.ress.2012.08.002