Méthodes de recherche opérationnelle pour la résolution de problèmes intégrés d'ordonnancement de la production et de distribution

Azeddine Cheref

UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale MIPTIS

LABORATOIRE D'INFORMATIQUE

THÈSE présentée par :

Azeddine CHEREF

soutenue le : 28 mars 2017

pour obtenir le grade de : Docteur de l'université François - Rabelais de Tours

Discipline/ Spécialité : Informatique

Méthodes de recherche opérationnelle pour la résolution de problèmes intégrés d'ordonnancement de la production et de distribution

THÈSE DIRIGÉE PAR :

ARTIGUES Christian	Directeur de recherche, CNRS-LAAS
BILLAUT Jean-Charles	Professeur, Université François - Rabelais de Tours

RAPPORTEURS :

CHRÉTIENNE Philippe	Professeur Émérite, Université Pierre et Marie Curie
RAPINE Christophe	Professeur, Université de Lorraine

JURY :

AGNETIS Alessandro	Professeur, Università degli Studi di Siena
ARTIGUES Christian	Directeur de recherche, CNRS-LAAS
BILLAUT Jean-Charles	Professeur, Université François - Rabelais de Tours
CHRÉTIENNE Philippe	Professeur Émérite, Université Pierre et Marie Curie
MOUKRIM Aziz	Professeur, Université de Technologie de Compiègne
RAPINE Christophe	Professeur, Université de Lorraine

Remerciements

Après ces trois années de thèse, je réalise que l'aboutissement de ce projet de recherche n'aurait pas pu se faire sans le soutien de nombreuses personnes. En effet, leur générosité ainsi que l'intérêt manifesté à l'égard de ma recherche m'ont permis de progresser durant ces années de thèse.

J'aimerais tout d'abord remercier mes encadrants de thèse : Jean-Charles Billaut et Christian Artigues. Merci pour le soutien continu, appuie et encouragements dont vous avez fait preuve ainsi que la liberté que vous avez pu me donner tout au long de ces années. Je vous remercie pour tous ces riches échanges scientifiques, mais aussi humains que nous avons pu partager.

J'ai également eu la chance de travailler avec Alessandro Agnetis, je te remercie pour les conseils et les commentaires donnés qui ont contribué à mener à bien mes travaux de recherche et à orienter mes réflexions.

Outre les personnes déjà citées, je tiens à adresser mes sincères remerciements aux autres membres de mon jury de thèse, les Professeurs Philippe Chrétienne, Aziz Moukrim et Christophe Rapine pour m'avoir fait l'honneur de juger mon travail.

Cette thèse a été réalisée entre les villes de Tours et Toulouse. Ainsi, je remercie mes collègues de l'équipe ROOT (ex-OC) du laboratoire d'informatique de Tours et ceux de l'équipe ROC du LAAS-CNRS de Toulouse de m'avoir permis d'évoluer professionnellement à leurs côtés. Merci aussi à mes collègues de l'administration et du service informatique de Polytech Tours.

J'aimerais adresser un merci tout particulier à Amer Soukhal pour sa disponibilité et ses nombreux conseils.

On arrive aux doctorants avec lesquels j'ai partagé de très bons moments et qui m'ont permis de décompresser. Merci à Boukhalfa, Frédéric, Gaëtan, Kairaba et tous mes nombreux autres amis doctorants de Tours et de Toulouse qui ont fait que ces trois années soient un plaisir.

Enfin un merci affectueux à toute ma famille qui m'a soutenu tout au long de mon parcours et sans laquelle rien ne serait possible, en particulier mes parents Mohand Said et Malika, mon frère Djamel, mes deux sœurs Nesrine et Kahina et ma femme Hanane. –

Résumé

La production et la distribution sont deux éléments essentiels dans une chaîne de production. En effet, dans de nombreux systèmes de production, les produits finis sont livrés de l'usine vers différents clients, entrepôts ou centres de distribution. Afin d'assurer une optimisation globale des performances, nous abordons dans cette thèse le problème intégré d'ordonnancement et de distribution. Dans la littérature, de nombreux articles traitent des approches intégrées, impliquant des décisions de production et de distribution à un niveau stratégique. De plus en plus d'articles abordent ces problèmes au niveau opérationnel. De nouveaux problèmes de cette catégorie sont abordés dans cette thèse. Dans un premier temps le problème déterministe est étudié, et plusieurs cas sont abordés. Pour chacun d'eux, une étude de complexité est proposée et des algorithmes de résolution exacte (programmation dynamique, programmation mathématique, génération de colonnes) sont proposés, ainsi que des méthodes approchées. L'incertitude sur les données est introduite dans la deuxième partie de la thèse et des méthodes exactes et heuristiques sont proposées pour résoudre des problèmes intégrés robustes d'ordonnancement et de distribution. En particulier, des approches de programmation linéaire en nombres entiers et des métaheuristiques de type tabou sont proposées pour résoudre les problèmes d'optimisation robuste standard et d'optimisation "robuste récupérable". Ces méthodes sont évaluées et les résultats montrent l'efficacité des méthodes que nous proposons.

Mots clés : Recherche opérationnelle, ordonnancement, tournées de véhicules, complexité, programmation dynamique, génération de colonnes, robustesse.

Abstract

Production and delivery are two essential elements in supply chain management. Indeed, in many production systems, finished products are delivered from the factory to different customers, warehouses or distribution centers. In order to ensure a global optimization of performances, we address in this thesis the integrated problem of scheduling and delivery. In the literature, a large number of papers deal with integrated approaches at a strategic level. More and more papers deal with these problems at an operational level. New problems in this category are addressed in this thesis. In a first part, the deterministic problem is studied and several cases are treated. For each of them, a study of complexity is proposed and exact (dynamic programming, mathematical programming, column generation) as well as approximated methods are proposed. Data uncertainty is introduced in the second part of the thesis and exact and heuristic methods are proposed to solve integrated problems of scheduling and delivery. In particular, integer linear programming approaches and tabu search heuristic are proposed to solve the standard robust approach and the "recoverable robustness" approach. These methods are evaluated and the results show the efficiency of the methods that we propose.

Keywords : Operations research, scheduling, vehicle routing, complexity, dynamic programming, column generation, robustness.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Dans une chaîne logistique, la production et la distribution sont interdépendantes, mais elles sont généralement traitées séparément. Cette approche non coordonnée facilite le traitement du problème mais peut conduire à des solutions sous-optimales. Par conséquent, pour atteindre une performance opérationnelle optimale dans une chaîne logistique, il est crucial d'intégrer ces deux fonctions afin de les planifier de manière coordonnée. En effet, la compétition que se livrent les entreprises et les défis auxquels elles font face rend le traitement de la production et la distribution de manière intégrée nécessaire dans de nombreuses situations pratiques.

Durant la thèse, nous avons été confrontés à une problématique réelle dans le contexte de la production et la distribution de préparations de chimiothérapies injectables. Au CHRU de Tours, une fois les poches ou seringues préparées, celles-ci sont emmenées dans les services (voire dans des services sur un autre site hospitalier de la ville) par une personne, chargée d'effectuer ces livraisons. Produire ces poches uniquement en fonction de leur destination permettrait sans doute d'optimiser la livraison, mais cela ne correspond pas nécessairement aux rendez-vous convenus avec les patients. Inversement, produire les poches en fonction des dates d'administration souhaitées afin de réduire l'attente des patients, sans tenir compte du lieu de la livraison n'est pas non plus totalement satisfaisant. Cette problématique se retrouve dans tout système où la production et la livraison doivent être décidées de façon coordonnée.

Durant cette thèse, nous avons traité le problème intégré de production et de distribution où la partie production consiste à trouver une séquence de production de tâches sur une machine et la partie distribution consiste à livrer les produits finis vers le(s) client(s). Après une étude de la littérature, nous avons considéré dans un premier temps le problème déterministe et proposé des études de complexité et des approches de résolution pour de nouveaux problèmes. Dans la seconde partie de la thèse, des incertitudes ont été introduites aux modèles d'ordonnancement et de tournées de véhicule et des approches de résolution ont été proposées. C'est ainsi que durant cette thèse, plusieurs angles de la recherche opérationnelle ont été abordés tels que des étude de complexité, les algorithmes de programmation dynamique, des algorithmes d'approximation, des méta-heuristiques, de la programmation linéaire en nombres entiers et de la génération de colonnes, ou encore de l'optimisation robuste.

1.1 Revue de la littérature

Dans cette section, nous examinons différentes représentations pour le problème d'ordonnancement basées sur une séquence de tâches et nous donnons une définition des problèmes de tournées de véhicules. Enfin, nous présentons plus en détail le problème intégré de production et de distribution ainsi que les différentes manières de l'aborder.

1.1.1 Problèmes d'ordonnancement

Un problème d'ordonnancement consiste à définir un ensemble de dates de début d'exécution pour un ensemble de tâches ayant des ressources communes. En prenant en compte certaines contraintes de temps (telles que les dates de début au plus tôt, etc.) et dans le but d'optimiser une fonction objectif, une solution à un problème d'ordonnancement est représentée par une séquence d'exécution d'un ensemble de tâches sur chaque machine (voir [Brucker, 2007] pour plus de précision sur les problèmes d'ordonnancement).

On retrouve les problèmes d'ordonnancement dans tous les types d'organisation. Cependant, le domaine d'application le plus célèbre reste l'industrie et plus précisément la production industrielle où les problèmes d'ordonnancement prennent une place importante dans la gestion de production. Les problèmes d'ordonnancement sont également présents dans des domaines tels que la gestion de projets, la gestion des emplois du temps, les systèmes informatique, etc. Récemment, d'autres domaines d'applications tels que le traitement des grandes masses de données [Berlińska et Drozdowski, 2015, Berlińska et Drozdowski, 2011] sont apparus.

La théorie de l'ordonnancement s'est développée conjointement avec la théorie de la complexité. Les méthodes utilisées pour résoudre les problèmes d'ordonnancement viennent du domaine de l'optimisation combinatoire. Ainsi, des méthodes exactes sont utilisées pour la recherche de solutions optimales et des méthodes approchées permettent de trouver des solutions aussi bonnes que possible.

On note $\mathcal{J}$ l'ensemble de n $(J_1, \ldots, J_n)$ travaux (tâches) à ordonnancer. Une tâche est représentée par une seule opération, ou par plusieurs opérations dans le cas d'un problème d'ordonnancement avec plusieurs ressources (machines) par exemple. Dans notre cas, on se place dans environnement à une machine. On note $\mathcal{M}$ la machine et on suppose que celle-ci est disponible à l'instant 0 pour l'exécution des différentes tâches. A chaque tâche J_j, on associe un temps d'exécution p_j sur la machine. On suppose que la préemption n'est pas autorisée (i.e., une tâche est exécutée sans interruption sur la machine).

La qualité d'un ordonnancement est donnée par une mesure de performance basée sur la fin d'exécution des tâches. Les fonctions objectif les plus communes sont la durée totale d'ordonnancement notée $C_{\max}$ avec $C_{\max} = \max_{J_j \in \mathcal{J}} C_j$ ainsi que la somme des dates de fin d'exécution des tâches $\sum_{J_j \in \mathcal{J}} C_j$. Si le travail J_j est supposé finir à la date d_j, appelée la date d'échéance de J_j, nous définissons par L_j le retard de J_j, avec $L_j = C_j - d_j$ et $L_{\max} = \max_{J_j \in \mathcal{J}} L_j$, aussi appelé le retard maximal, est une autre mesure de performance importante pour le problème d'ordonnancement. D'autres mesures de performance peuvent être définies pour l'ordonnancement (voir [Brucker, 2007] ou [Błażewicz *et al.*, 2015] pour plus de détails). Un exemple d'ordonnancement sur un environnement à une machine est

illustré dans l'exemple suivant.

Exemple : environnement à une machine

Soient un ensemble de $n = 5$ tâches à ordonnancer sur une seule machine. A chaque tâche J_j sont associées : une durée d'exécution p_j, une date de fin souhaitée d_j, et une date de début au plus tôt r_j. L'objectif est de trouver une séquence d'exécution des tâches de façon à minimiser le retard maximal L_{max}. Les données du problème sont indiquées dans le tableau suivant.

J_j	J_1	J_2	J_3	J_4	J_5
r_j	0	7	3	4	3
p_j	3	4	1	2	4
d_j	3	14	4	6	10

Il faut noter que ce problème est NP-difficile au sens fort, i.e. il n'existe pas d'algorithme polynomial pouvant résoudre ce problème à l'optimum, a moins que $P = NP$ [Garey et Johnson, 1979]. On suppose dans un premier temps que les tâches sont ordonnancées par ordre croissant de leurs dates de début au plus tôt r_j (Earliest Release Date). Soit σ_{ERD} la séquence obtenue. Supposons maintenant que les tâches soient ordonnancées selon un ordre croissant des dates de fin souhaités d_j (Earliest Due Date first). Soit σ_{EDD} la séquence obtenue. Une représentation sous forme de diagramme de Gantt ainsi que l'évaluation des deux séquences sont données dans la Figure 1.1.

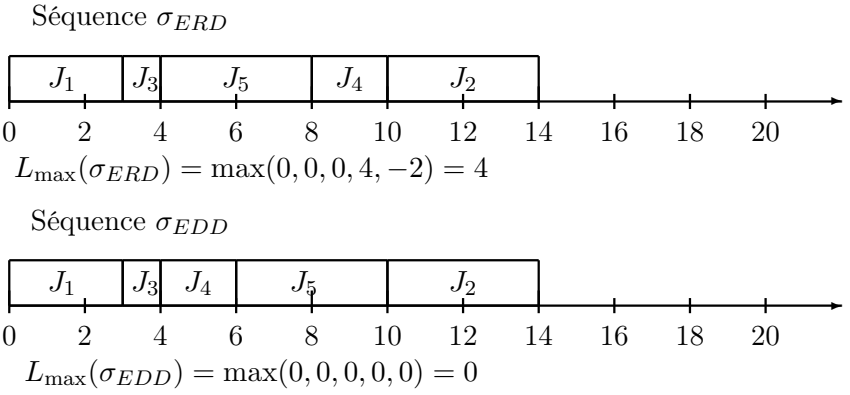


FIGURE 1.1 – Représentation des séquences σ_{ERD} et σ_{EDD} sous forme de diagramme de Gantt et leurs évaluations.

1.1.2 Problèmes de tournées de véhicules

Le problème de tournées de véhicules (VRP : Vehicle Routing Problem) est un problème d'optimisation combinatoire qui cherche à satisfaire la demande d'un ensemble de clients par une flotte de véhicules. Le VRP a été introduit pour la première fois en 1959 par Dantzig et Ramser afin de résoudre le problème de livraison d'essence aux stations service. Depuis lors, l'intérêt pour le VRP a évolué d'un petit groupe de mathématiciens à une

large panel de chercheurs et de praticiens de différentes disciplines et constitue aujourd'hui encore l'un des champs les plus importants des problèmes de transport, de distribution et de logistique. Le VRP est une appellation générique qui englobe toute une classe de problèmes dans lesquels un ensemble de routes empruntées par une flotte de véhicules partant d'un ou plusieurs dépôts doit être déterminé afin de satisfaire la demande d'un ensemble de clients situés à différentes localisations. L'objectif des problèmes de tournées de véhicules est de minimiser le coût de transport total. La majorité des problèmes réels de VRP sont plus complexes que le VRP classique. Ainsi en pratique, des contraintes de capacité du véhicules, de fenêtres de temps durant lesquels un client dois être livré, etc sont ajoutées au VRP classique. Le VRP est un problème d'optimisation combinatoire NP-difficile au sens fort et seules de petites instances peuvent être résolues à l'optimum par une méthode exacte. La figure 1.2 montre un exemple de problème ainsi qu'une des solutions possibles. Le lecteur trouvera un état de l'art récent des problèmes de tournées de véhicules dans [Toth et Vigo, 2014].

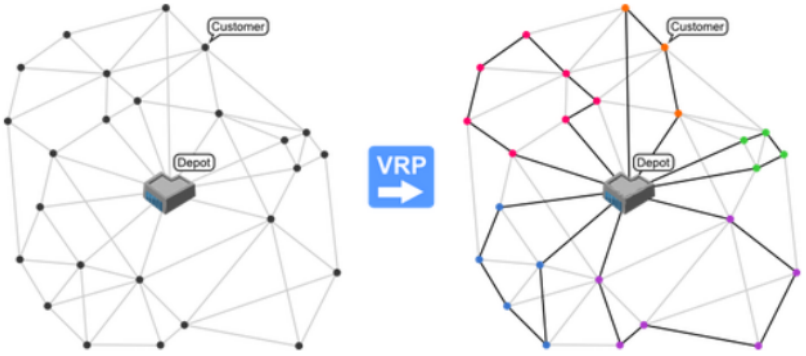


FIGURE 1.2 – Une instance de VRP (à gauche) et sa solution (à droite). Source [NEO, 2017]

1.1.3 Problèmes intégrés de production et de distribution

Dans une approche non intégrée du problème où les parties production et distribution sont traitées séparément, la partie production peut être traitée au préalable et une séquence d'ordonnancement est alors déterminée. Une séquence de production et, par conséquent, les dates de fin de production des tâches correspondantes constituent alors les données d'entrée du problème de tournées de véhicules et une séquence de livraison des tâches est alors recherchée afin de livrer les tâches vers les différents clients. Ainsi, les dates de fin de production des tâches qui représentent les dates de disponibilités des tâches pour la livraison peuvent être vues comme étant des dates de début au plus tôt (release dates) des tournées dans un problème de tournées de véhicules. Ce problème est traité par [Cattaruzza *et al.*, 2016], parmi d'autres références. Le cas où la partie distribution est traitée en premier est également possible. Dans ce cas, le problème d'ordonnancement prend alors comme données d'entrée les dates de départ des tournées ainsi que leur constitution et celles-ci peuvent alors être considérées comme des dates de fin de production au plus

tard des tâches (due dates).

Dans cette section, on aborde de façon conjointe un problème d'ordonnancement de production et un problème de distribution.

Dans la majorité des travaux réalisés sur les problèmes intégrés d'ordonnancement et de distribution, la production est effectuée sur un environnement de production à une seule machine. Nous présentons dans ce qui suit un exemple dans lequel la production se fait sur une machine et les produits finis sont livrés vers un ensemble de clients situés à différents endroits.

Exemple : problème intégré de production et de distribution

Soit un ensemble de $n = 7$ tâches à produire sur une machine et à livrer par un véhicule de capacité $c = 3$. Une fois les tâches produites, elles sont livrées vers un ensemble de 7 clients situés à des endroits différents. La tâche J_j est à livrer au site j.

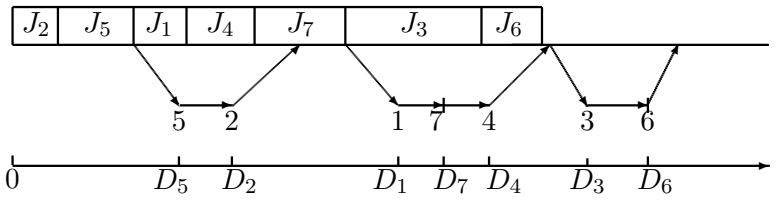


FIGURE 1.3 – Illustration d'une solution d'un problème intégré de production et de distribution

La figure 1.3 montre une solution du problème intégré de production et de livraison dans laquelle la séquence de production est $(J_2, J_5, J_1, J_4, J_7, J_3, J_6)$ et le séquence de livraison est $(J_5, J_2, J_1, J_7, J_4, J_3, J_6)$. Ainsi, dans la solution illustrée, le véhicule part livrer les tâches $\{J_2, J_5\}$ à la date de fin de production de la tâche J_5 et livre J_5 en premier à la date D_5 puis J_2 à la date D_2. Au retour du véhicule au dépôt, le véhicule attend la fin de production de la tâche J_7 et part livrer le lot (batch) de tâches $\{J_1, J_4, J_7\}$. Les tâches restantes J_3, J_6 étant prêtes, le véhicule repart immédiatement pour les livrer.

Après cette introduction de la problématique, la littérature existante sur le problème de production et de distribution intégrés est passée en revue et les caractéristiques des problématiques ainsi que les approches de résolution qui les accompagnent sont discutées dans la section qui suit.

1.2 Etat de l'art général

Au cours des vingt dernières années, un nombre croissant de recherches ont été consacrées à la mise au point de modèles intégrés de production et de distribution. L'état de l'art de Chen [Chen, 2010b] permet d'avoir une très bonne vision de la littérature sur cette catégorie de problèmes. Dans la grande majorité des articles de la littérature, la coordination de la production et de la distribution est assurée par la création de lots, à savoir, les produits d'un même lot sont embarqués ensemble et livrés à leurs destinations respectives au cours d'un seul voyage. Lors de la formation des lots, plusieurs paramètres doivent être pris en compte tels que le temps de traitement pour la partie ordonnancement et la loca-

lisation des clients pour la partie livraison. Un nouvel état de l'art est également proposé par Moons, Ramaekers, Caris et Arda [Moons *et al.*, 2016] sur les problèmes intégrés de production et de distribution. Cependant, les auteurs excluent certains cas très étudiés dans la littérature tel que celui où les tâches sont livrées vers un seul site (client) et se concentrent sur les études incluant de manière explicite le problème de tournées de véhicules. Cette distinction s'explique par le fait qu'un grand nombre d'articles dans lesquels la partie tournées de véhicules est traitée de manière explicite est récemment paru.

Nous présentons dans cette section les travaux les plus pertinents dans cette catégorie de problèmes afin de donner une vision globale des différentes caractéristiques des problèmes étudiés. Ainsi, la partie production des articles que nous traitons dans cette section peut présenter plusieurs caractéristiques différentes telles que le nombre de machines, des temps et des coûts de production spécifiques, des temps de maintenance, etc. Des caractéristiques particulières sont également considérées dans la partie distribution telles que le nombre de véhicules disponibles pour effectuer la livraison ainsi que leurs propriétés (ex. capacité), les données du transport telles que le temps et le coût, des restrictions sur le nombre de tournées qu'un véhicule peut effectuer et sur les dates de livraison des produits, etc.

Etant donné que le cas où le problème de tournées de véhicules ne fait pas partie de la décision a également été traité durant cette thèse, beaucoup d'articles auxquels nous nous référons traitent le problème intégré d'ordonnancement et de livraison en considérant que les produits finis sont livrés vers un unique client (entrepôt) ou que la séquence de livraison est prédéfinie. Ainsi dans certains des premiers travaux sur cette catégorie de problèmes, les temps de transport ne sont pas pris en compte et le problème revient à constituer des lots de tâches à livrer ensemble. Un tel problème est traité par Hal et al [Hall *et al.*, 2001] qui considèrent des dates de départ du véhicule fixes et un environnement de production à une ou plusieurs machines ainsi que plusieurs critères d'optimisation tirés des problèmes d'ordonnancement.

Un nombre important de travaux traitent également le problème dans lequel la partie tournées de véhicules ne fait pas partie de la décision. En effet, les cas où les produits finis sont livrés vers un petit nombre de clients et ceux dans un environnement à plusieurs machines où les tâches sont transportées entre différentes machines constituent une part importante de la littérature dans ce domaine. Dans [Lee et Chen, 2001] les deux situations sont considérées dans la partie distribution. Dans la première, les tâches sont d'abord produites sur la première machine puis transportées vers une deuxième machine afin de terminer la production. Le second type consiste à transporter les tâches produites vers un unique client (entrepôt). Plusieurs problèmes sont traités et des preuves de complexité ainsi que des algorithmes sont proposés. Dans [Chang et Lee, 2004], les auteurs considèrent la partie production comme étant un problème d'ordonnancement à une machine ou à deux machines parallèles et considèrent deux cas pour la partie distribution. Un premier cas dans lequel les tâches sont livrées vers un unique client (site) et un cas où les tâches sont livrées vers deux sites distincts. La livraison est effectuée par un seul véhicule avec une capacité limitée dans lequel chaque tâche occupe un espace particulier. Les auteurs proposent des preuves de complexité et des algorithmes d'approximation pour la minimisation de la date de retour du véhicule au dépôt après que toutes les tâches aient été livrées. Une extension au problème d'ordonnancement à une machine et de livraison vers un unique client traité par [Chang et Lee, 2004] est traitée par [Chen, 2010a] où un

ensemble de véhicules est disponible pour effectuer la livraison et les temps de transport entre le site de production et le client dépendent du véhicule choisi. Les auteurs proposent un programme linéaire en nombres entiers pour résoudre le problème. Ce problème est également traité par [He *et al.*, 2006] et [Zhong *et al.*, 2007]. Dans [He *et al.*, 2006], les auteurs proposent une amélioration à l'une des heuristiques d'approximation proposées dans [Chang et Lee, 2004] et traitent également le cas où des temps de maintenance sur la machine sont pris en compte et proposent un algorithme polynomial en le nombre de tâches. Pour le problème à une machine et un site le livraison, [Zhong *et al.*, 2007] proposent la meilleure approximation possible améliorant ainsi les résultats proposés dans [Chang et Lee, 2004] et [He *et al.*, 2006]. Les auteurs proposent également une heuristique procurant de meilleures performances pour le problème à deux machines parallèles traité par [Chang et Lee, 2004]. Dans [Dong *et al.*, 2013], l'environnement de production est représenté par un problème d'ordonnancement de type cheminement libre (open shop) à deux machines dans lequel les tâches produites sont livrées vers un unique client par un véhicule de capacité finie. Les auteurs proposent une heuristique avec garantie de performance pour le cas général. Les auteurs traitent également le cas particulier dans lequel le véhicule livre une seule tâche à la fois.

Le problème de tournées de véhicules étant NP-difficile au sens fort, un nombre important d'articles dans la littérature traitent en même temps les cas particuliers où les tâches sont livrées vers un seul client et ceux où elles sont livrées vers plusieurs clients. En effet, la complexité de ces cas particuliers a priori simples est étudiée et des algorithmes d'approximation avec garantie de performance sont fréquemment proposés. Dans [Chen et Vairaktarakis, 2005], les auteurs traitent plusieurs cas particuliers du problème d'ordonnancement et de livraison. Les tâches sont exécutées sur une seule machine ou un ensemble de machine parallèles puis livrées vers un seul client ou un ensemble de clients par un ensemble suffisant de véhicules homogènes de capacité limitée. L'objectif à minimiser comprend les coûts de transports qui comprennent un coût fixe d'utilisation des véhicules et les coûts de transport entre les différentes destinations. La fonction objectif prend également en compte les dates de livraison des produits. Ainsi les auteurs considèrent séparément la date moyenne de livraison des produits (correspondant à la somme des dates de livraisons) et la date de livraison de la dernière tâche. Les auteurs traitent six cas particuliers et pour chacun d'eux, un algorithme polynomial de programmation dynamique ou une heuristique avec garantie de performance sont proposés. Dans [Chen et Pundoor, 2006], les auteurs considèrent le problème traité par [Chen et Vairaktarakis, 2005] et traitent le cas où les tâches sont exécutées sur des machines situées à différents endroits et livrées vers un unique client. Des preuves de complexité ainsi que algorithmes de résolution sont proposés pour une multitude de cas. Les mêmes auteurs traitent cette fois dans [Chen et Pundoor, 2009] le problème d'ordonnancement à une machine et de distribution vers un client par un ensemble de véhicules homogènes. Les temps de transports sont supposés nuls et la capacité des véhicules limitée représente un poids maximum pouvant être transporté en même temps. Pour chaque tâche, des dates de fin de production au plus tôt ainsi qu'un poids occupé dans le véhicule sont introduits. Les auteurs traitent quatre cas du problème consistant à autoriser ou à interdire la préemption dans la production et la livraison. Pour chacun des problèmes considérés, les auteurs cherchent à minimiser le nombre de lots en faisant en sorte soit de livrer les tâches avant leurs dates de livraison souhaitées, soit que la moyenne

des dates de livraison soit inférieure à une durée donnée. Pour chacun des problèmes, les auteurs proposent des preuves de complexité ainsi que des algorithmes de résolution. Dans [Lu *et al.*, 2008], les auteurs considèrent le problème d'ordonnancement à une machine dans lequel les dates de début au plus tôt des tâches sont prises en compte. Les tâches produites sont livrées vers un unique client (site) par un véhicule de capacité finie représentant un nombre maximum de tâches pouvant être transportées dans la même tournée. L'objectif à minimiser étant la date de retour du véhicule au site de production après que toutes les tâches aient été livrées, les auteurs proposent un algorithme polynomial pour le cas où la préemption est autorisée. Dans le cas où la préemption n'est pas autorisée, les auteurs montrent que le problème est NP-difficile et proposent une heuristique avec garantie de performance. Dans [Condotta *et al.*, 2013], les dates de début de production au plus tôt sont également prises en compte et des dates de livraison au plus tard sont introduites au modèle. Les auteurs considèrent un ensemble de tâches à produire sur une machine et à livrer vers un seul client par un ensemble fini de véhicules identiques de capacité limitée. Un programme linéaire en nombres entiers ainsi qu'une heuristique tabou sont proposés pour la résolution du problème. Dans [Li *et al.*, 2005], les auteurs considèrent un problème d'ordonnancement à une machine et de livraison et cherchent à minimiser la somme des dates de livraison des tâches. Le problème est montré NP-difficile pour un nombre de sites quelconque et proposent un algorithme polynomial de programmation dynamique pour la cas où les produits finis sont livrés vers une seule destination. Les auteurs proposent également un algorithme de programmation dynamique polynomial pour le cas où le nombre de sites est fixe. Un environnement de production spécifique appelé "*bundling operations*" est considéré par Li et Vairaktarakis [Li et Vairaktarakis, 2007], dans lequel une tâche se compose de deux opérations différentes. Ainsi, les deux opérations d'une tâche sont exécutées sur deux machines indépendamment l'une de l'autre et une fois les deux opérations terminées, la tâche est transportée vers un unique client par un ensemble de véhicules de capacités limitées. Les coûts de transport étant pris en compte, l'objectif est de minimiser le coût de livraison total plus une somme pondérée des dates de livraison . Les auteurs proposent des heuristiques polynomiales ainsi que des algorithmes de programmation dynamique basés sur des schémas d'approximation.

L'acheminement des matières premières aux sites de production est considéré dans certains travaux de la littérature. Ainsi, Li et Ou dans [Li et Ou, 2005] considèrent un problème d'ordonnancement à une seule machine et deux différents entrepôts situés à deux endroits différents. Les tâches à produire se trouvent au préalable dans le premier entrepôt puis acheminées vers le site de production. Une fois la production d'une tâche terminée, celle-ci peut être livrée vers le second entrepôt. Le transport est effectué par unique véhicule dont la capacité dépend du type de tâches transportées (finies ou pas). Les auteurs cherchent à minimiser la date de fin de livraison de toutes les tâches. Les auteurs donnent une preuve de complexité pour le cas général, proposent un algorithme polynomial pour le cas où la séquence de production est fixée ainsi que des heuristique avec garantie de performances pour le cas général et des cas particuliers. Ce modèle est étendu dans [Wang et Cheng, 2009] au cas où le transport est effectué par deux véhicules. Ainsi, un premier véhicule transporte les tâches à exécuter au site de production (machine) et un second véhicule transporte les produits finis du site de production jusqu'au deuxième entrepôt. Les auteurs proposent une heuristiques avec garantie de performance ainsi que

des algorithmes polynomiaux pour plusieurs cas particuliers.

Il existe également quelques travaux sur la production et la livraison de produits périssables. Dans [Armstrong *et al.*, 2008], les auteurs traitent le problème à séquence fixée de production à une machine et de livraison vers ensemble de clients de produits dont la durée de vie est limitée. La livraison est réalisée en une seule tournée et par un unique véhicule et des fenêtres de temps pour la livraison sont prises en compte. La séquence étant prédéfinie, le problème consiste à déterminer l'ensemble de tâches à produire et livrer afin de maximiser la satisfaction de la demande. Les auteurs proposent une heuristique et un algorithme de séparation et évaluation (branch & bound) pour résoudre le problème. Dans [Viergutz et Knust, 2014], les auteurs soulignent une erreur dans l'algorithme de séparation et évaluation proposé par [Armstrong *et al.*, 2008]. Les auteurs proposent une correction au séparation et évaluation et étendent les travaux proposés par [Armstrong *et al.*, 2008] au cas où des temps d'attente sur la machine sont autorisés. Des heuristiques sont également proposées pour le cas où la séquence de production et de livraison ne sont pas fixées. Dans [Geismar *et al.*, 2008], les auteurs considèrent que la production se fait sur une machine où une quantité fixe est produite par unité de temps. Les produits sont considérés périssables et doivent être livrés par un unique véhicule de capacité limitée vers un ensemble de clients situés dans différents sites. Le problème consiste à déterminer les clients à livrer dans chaque tournée de sorte que la date de péremption ne soit pas dépassée et en minimisant la date de fin de livraison de tous les produits. Les auteurs montrent que le problème est NP-difficile et proposent des bornes inférieures de l'objectif ainsi qu'une heuristique à deux phases pour résoudre le problème. Le même problème est étudié par [Karaoğlan et Kesen, 2017] et [Devapriya *et al.*, 2016] pour lequel les premiers auteurs proposent une procédure de séparation et génération de coupes et les seconds proposent un algorithme génétique ainsi que deux algorithmes mémétiques.

Une contrainte particulière qui stipule que le véhicule doit partir exactement à la date de fin de production des tâches transportées durant une tournée est introduite dans [Gao *et al.*, 2015]. Les auteurs considèrent le problème d'ordonnancement à une machine et de livraison où un seul véhicule de capacité limitée est disponible pour livrer les tâches finies. Après une étude de complexité montrant que le problème est NP-difficile, les auteurs proposent un algorithme polynomial pour le cas particulier dans lequel les temps de transports sont identiques et les durées de production des tâches sont identiques. Une heuristique polynomiale avec garantie de performance est également proposée pour le cas général. Dans [Lee *et al.*, 2014], les auteurs considèrent un problème de production et de livraison de substances médicales radioactives. La production est réalisée sur un ensemble de machines et chaque tâche est exécutée sur une seule machine. Les produits finis sont livrés vers les clients par un ensemble de véhicules. Le problème comprend plusieurs contraintes et paramètres tels que les coûts et la vitesse de production, les coûts de transports, les fenêtres de temps pour la livraison, etc. L'objectif est de minimiser un coût total incluant les coûts de production, les coûts de transport et les coûts fixes d'utilisation d'un véhicule. Les auteurs proposent un programme linéaire en nombre entiers ainsi qu'un algorithme de recherche à voisinage étendu (large neighborhood search).

Une caractéristique particulière est introduite par [Ullrich, 2013] dans la partie distribution. Les auteurs considèrent un environnement à machines parallèles avec des temps de disponibilité sur les machines et un problème de tournées de véhicules pour la partie

distribution. La livraison est réalisée par un ensemble de véhicules de capacité finie et des temps de disponibilité des véhicules sont introduits et pour chaque tâche, des dates de fin de production au plus tard ainsi que des fenêtres de temps et l'objectif est de minimiser la somme des retards de production pour la partie production et des retards de livraison pour le problème de tournées de véhicules. Les auteurs proposent dans un premier temps un programme linéaire en nombres entiers pour le problème d'ordonnancement, pour le problème de tournées de véhicules ainsi que pour le problème intégré d'ordonnancement et de tournées de véhicules. Les auteurs proposent également deux heuristiques de décomposition ainsi qu'un algorithme génétique. Dans [Lee, 2015], l'auteur traite le problème intégré d'ordonnancement et de distribution dans lequel des coûts de transport dépendent de la date de livraison. La production est représentée par un problème d'ordonnancement à une machine dans lequel des temps d'arrêt de production sont permis et un grand ensemble de véhicules est disponible pour effectuer la livraison. L'auteur cherche à minimiser la somme du coût de l'ordonnancement et du coût de livraison. Plusieurs objectifs sont pris en compte dans la partie ordonnancement tels que la somme des dates de fin de production des tâches, le retard maximum, etc. L'auteur étudie la complexité en proposant soit des preuves de NP-difficulté ou bien des algorithmes polynomiaux de programmation dynamique pour une variété de fonctions objectifs et de cas particuliers. Dans [Scholz-Reiter *et al.*, 2010], les auteurs considèrent un contexte de chaîne de production globale et se concentrent sur l'interface entre la production et la partie logistique le long de la chaîne d'approvisionnement. La partie production consiste à produire des tâches sur un ensemble de machines de type cheminement unique hybride (hybrid flowshop) et de les affecter ensuite à des tournées. L'objectif est une fonction comprenant plusieurs paramètres tels que les coûts de production, les coûts de stockage, les coûts des tournées, etc. Les auteurs proposent une formulation mathématique testée sur de petites instances.

1.3 Organisation du manuscrit

Ce manuscrit est divisé en deux parties et est organisé de la manière suivante.

Dans la partie I, le problème déterministe de production sur une machine et de distribution intégré est abordé. Dans le chapitre 2, les séquences de production et de distribution sont considérées fixées. Nous montrons dans un premier temps que le problème est NP-difficile puis nous présentons un algorithme de programmation dynamique pseudo-polynomial pour le résoudre. Plusieurs cas sont traités et, pour chacun d'eux, le problème est soit prouvé NP-difficile soit un algorithme de programmation dynamique polynomial est proposé pour sa résolution. Dans le chapitre 3, nous considérons que les tâches sont produites sur une machine et livrées vers un seul client et que le véhicule part effectuer les livraisons à des dates fixes. Le problème est montré NP-difficile et un algorithme d'approximation est proposée pour un cas particulier. Un modèle de programmation linéaire en nombres entiers et une formulation étendue adaptée à la génération de colonnes sont proposés pour le cas général. Dans le chapitre 4, le problème intégré d'ordonnancement et de tournées de véhicules est abordé. Après une brève étude de complexité de cas particuliers supplémentaires, des formulations étendues adaptées à la génération de colonnes sont proposées pour deux fonctions objectifs.

Dans la partie II, l'incertitude sur les données est introduite et le problème intégré de production et de distribution robuste est traité. Ainsi, une définition de la robustesse et les différentes manières de l'aborder seront présentées dans le chapitre 5. Ensuite, nous abordons dans le chapitre 6 le problème d'ordonnancement robuste (sous la forme classique et sous la forme de robustesse récupérable) pour lequel nous proposons des modèles linéaires en nombres entiers et une heuristique tabou. Dans le chapitre 7, le problème de tournées de véhicules est introduit à l'ordonnancement et des modèles PLNE ainsi que des heuristiques tabou sont proposés pour la résolution du problème intégré d'ordonnancement et de tournées de véhicules robuste également sous les deux formes considérées. Enfin dans le chapitre 8, des expérimentations sont effectuées et les résultats des différents algorithmes sont présentés. Ces résultats montrent l'intérêt de la robustesse récupérable par rapport à la robustesse classique.

Première partie

Production et distribution intégrées - cas déterministe

Chapitre 2

Problème d'ordonnancement et de livraison intégrés à séquence fixée et autres cas particuliers

Dans ce chapitre, on s'intéresse au cas particulier du problème d'ordonnancement à une machine et de livraison intégré dans lequel l'ordre d'exécution des tâches sur la machine et l'ordre de livraison sont les mêmes et prédéfinis. Les tâches sont livrées vers différents clients par un unique véhicule disposant d'une capacité limitée. Chaque tâche nécessite un temps d'exécution sur la machine et les temps de transport entre les clients sont également pris en compte. Étant donné que la séquence de production ainsi que l'ordre de livraison sont connus, le problème consiste alors à regrouper les tâches en lots à livrer durant la même tournée et notre objectif est de minimiser la somme des dates de livraison. Après une brève description du problème et un rappel de notations, une preuve de complexité ainsi qu'un algorithme de programmation dynamique pseudo-polynomial sont donnés. Dans la suite du chapitre, nous traitons quelques cas particuliers et pour chacun d'eux, nous établissons soit une preuve de complexité soit un algorithme polynomial pour sa résolution. Ce travail est issu d'une collaboration avec Alessandro Agnetis. Il a été présenté dans la conférence internationale avec actes ICORES [Cheref *et al.*, 2016a] et a été soumis à Journal of Scheduling (révision mineure).

2.1 Introduction et présentation du problème

Nous donnons ici quelques notations utilisées dans ce manuscrit avant de passer à la description du problème :

- n : nombre de tâches
- $\{J_1, J_2, \ldots, J_n\}$: ensemble des tâches à livrer respectivement aux sites $\{1, 2, \ldots, n\}$
- c : la capacité du véhicule
- $t_{i,j}$: le temps de transport entre le site i et le site j
- C_j : la date de fin d'exécution de la tâche J_j sur la machine. Cette date correspond à la date à laquelle la tâche J_j est disponible pour être livrée.

— D_j : la date de livraison de la tâche J_j.

La capacité c du véhicule est considérée dans ce chapitre comme étant le nombre maximal de tâches qu'il peut charger et par conséquent livrer durant une tournée. Ceci revient à dire que la taille s_j d'un job J_j est égale à 1 pour tout j. Les tâches doivent être livrées dans l'ordre dans lequel elles ont été produites et ainsi la séquence de production spécifie aussi l'ordre dans lequel les clients sont atteints. La séquence de production étant supposée fixée dans ce chapitre, le problème intégré d'ordonnancement et de distribution considéré dans ce chapitre se réduit à une partition des tâches en lots (i.e. un *schéma de batching*) dans lequel chaque lot est livré suivant l'ordre d'exécution des tâches sur la machine.

En général, la performance d'un système dépend de plusieurs décisions prises simultanément : ordonnancement de la production, constitution des lots et de la tournée de véhicules. Ceci requiert un *modèle intégré* permettant la coordination de tous ces aspects. Une solution pour notre problème avec séquence prédéfinie est une spécification d'un *schéma de batching* et constitue donc une variante très simplifiée d'un problème intégré de production et de distribution.

La mesure de performance que nous considérons dans ce chapitre dépend exclusivement des dates de livraison. En particulier, celle-ci est notée Z et a une des deux valeurs suivantes :

— la somme des dates de livraisons, i.e., $Z = \sum_{j=1}^{n} D_j$

— la somme de fonctions individuelles des dates de livraison, i.e., $Z = \sum_{j=1}^{n} f_j(D_j)$, où $f_j(D_j)$ est une fonction croissante des D_j

A noter que le seconde fonction inclut les fonctions telles que la somme (pondérée) des dates de livraison, la somme (pondérée) des retards, etc.

Le problème considéré peut être décrit par la notation $\alpha|\beta|\gamma$ due à [Lee et Chen, 2001]. En effet, une seule machine est utilisée pour la production et les tâches sont distribuées vers les différents clients ($1 \rightarrow D$), les tâches sont livrées vers n sites ($k = n$), la livraison est effectuée par un unique véhicule ($v = 1$) de capacité finie c. Enfin, la séquence de production et de livraison sont identiques et fixées (***fixed-seq***) et l'objectif est de minimiser la somme des dates de livraisons. Le problème est alors noté $1 \rightarrow D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$.

Problème $1 \rightarrow D|k = 1, v = 1, c,$ ***fixed-seq***$|\sum D_j$**.** *Etant donné n tâches avec des durées d'exécution p_j, $j = 1, \ldots, n$, des temps de transport $t_{i,j}$ pour tout $(i,j)_{i=1,\ldots,n;j=1,\ldots,n;i\neq j}$, et une séquence σ des n tâches. Trouver une partition des tâches en lots telle que Z minimum.*

Exemple : On considère une instance avec $n = 7$ tâches, une capacité du véhicule $c = 3$ et une séquence de production et de livraison $\sigma = (J_1, J_2, J_3, J_4, J_5, J_6, J_7)$. Etant donné que la séquence est fixée, seuls les temps de transport entre des sites successifs et entre le dépôt et les différents sites sont nécessaires. Le temps d'exécution des tâches, le temps de transport entre des sites successifs ainsi que les temps de transport du dépôt aux différents sites sont donnés dans le tableau suivant.

J_j	1	2	3	4	5	6	7
p_j	6	10	7	9	12	18	8
t_{jj+1}	7	-	6	6	-	8	-
t_{Mj}	6	9	7	-	8	5	-

La figure 2.1 montre une solution du problème dans laquelle le véhicule part à la date $C_2 = 16$ pour livrer le premier lot constitué des tâches $\{J_1, J_2\}$ et retourne au dépôt à la date 38. Le véhicule attend la fin de production de la tâche J_5 afin de livrer le lot $\{J_3, J_4, J_5\}$ et retourne au dépôt à la date 71. La production du lot $\{J_6, J_7\}$ étant terminée au retour du véhicule, celui-ci repart immédiatement afin de le livrer. Les dates de livraison des tâches $\{J_1, \ldots, J_7\}$ sont alors égales à $(22, 29, 51, 57, 63, 76, 84)$ respectivement et la somme des dates d'arrivées $\sum_{j=1}^{7} = 382$.

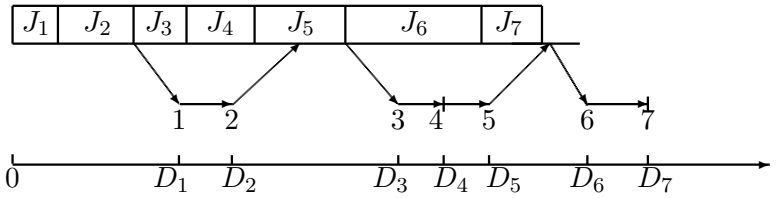


FIGURE 2.1 – Une illustration du problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$

Dans ce chapitre, nous nous concentrons principalement sur le problème d'optimisation $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$ à savoir, comment former les lots avec une séquence de production donnée. Le problème général dans lequel la séquence de production et de distribution n'est pas prédéfinie et doit être décidée est traité dans le chapitre 4. Celui-ci a par ailleurs été prouvé NP-difficile par [Li *et al.*, 2005] par une réduction polynomiale au *Traveling Repairman Problem.*

Nos contributions dans ce chapitre sont comme suit. Dans un premier temps, nous caractérisons complètement la complexité du problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$. Ainsi, pour une fonction objectif Z de type somme pondérée des dates de livraison, nous montrons que le problème est NP-difficile au sens ordinaire et qu'il peut être résolu en temps pseudo-polynomial pour n'importe quelle fonction de dates de livraison. Par la suite, nous élargissons la preuve de complexité au problème dans lequel plusieurs produits doivent être livrés vers un même client ainsi qu'au cas où la préemption dans la livraison est permise. Enfin, nous nous intéressons à quelques cas particuliers tels que des temps de transport identiques entre les clients, un nombre de sites borné et le cas où les temps d'exécution sont nuls. Pour tous ces cas, des algorithmes de programmation dynamique polynomiaux sont donnés.

2.2 Complexité du problème $1 \to D|k, v = 1, c,$ ***fixed-seq***$|\sum f_j D_j$

Du moment que l'ordre de production est prédéfini et que les produits sont livrés vers les clients en suivant le même ordre, nous supposons que la séquence de production σ est comme suit $\sigma = (J_1, J_2, ..., J_n)$ et que les tâches sont livrées dans le même ordre c'est-à-dire $(L_1, L_2, ..., L_n)$. Ceci implique aussi que seuls les temps de transport entre deux sites successifs $t_{j,j+1}$ et les temps de transport entre les sites et le dépôt $t_{j,M} = t_{M,j}$ présentent un intérêt.

Considérons d'abord le problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$ où la fonction objectif Z est la durée totale de livraison. Afin d'entamer notre preuve de complexité, nous introduisons le problème suivant :

Even-Odd Partition (EOP). Etant donné un ensemble de n paires d'entiers positifs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, où, pour tout i, $a_i > b_i$. Soit $K = \sum_{i=1}^{n}(a_i + b_i)$, existe-t-il une partition $(S, \bar{S})$ de l'ensemble des indices $I = \{1, 2, \ldots, n\}$ telle que

$$\sum_{i \in S} a_i + \sum_{i \in \bar{S}} b_i = K/2 \ ? \tag{2.1}$$

EOP est NP-difficile au sens ordinaire [Garey *et al.*, 1988]. Pour la preuve, nous allons utiliser une version légèrement modifiée du problème EOP.

Modified Even-Odd Partition (MEOP). Etant donné un ensemble de n paires d'entiers positifs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, où, pour tout i, $a_i > b_i$. Soit $Q = \sum_{i=1}^{n}(a_i - b_i)$, existe-t-il une partition $(S, \bar{S})$ de l'ensemble des indices $I = \{1, 2, \ldots, n\}$ telle que

$$\sum_{i \in S}(a_i - b_i) = Q/2 \ ? \tag{2.2}$$

A noter que les problèmes EOP et MEOP sont équivalents. En effet, supposons que pour une instance donnée, EOP admet une partition $(S, \bar{S})$. Pour la même instance, le problème MEOP admet une même partition $(S, \bar{S})$. Effectivement, en soustrayant $\sum_{i=1}^{n} b_i = \sum_{i \in S} b_i + \sum_{i \in \bar{S}} b_i$ des deux côtés de (2.1), on obtient :

$$\sum_{i \in S}(a_i - b_i) = K/2 - \sum_{i=1}^{n} b_i \tag{2.3}$$

Maintenant, d'après la définition de K et de Q, il se trouve que

$$Q = K - 2\sum_{i=1}^{n} b_i$$

Alors, nous pouvons voir que (2.3) est équivalent à (2.2) et ainsi nous pouvons présenter le résultat suivant.

Théorème 2.2.1 $1 \to D|k = n, v = 1, c,$ fixed-seq$|\sum_{j=1}^{n} D_j$ *est NP-difficile même si la capacité du véhicule* $c = 2$.

Démonstration. Nous pouvons commencer par dire que ce problème est évidement dans la classe NP et que cette preuve consiste à monter que ce problème est NP-difficile. En l'occurrence, NP-difficile au sens ordinaire par réduction polynomiale au problème MEOP.

Etant donnée une instance de MEOP, nous construisons une instance du problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$ comme suit :

- Un ensemble de $3n + 3$ tâches $\{J_1, J_2, \ldots, J_{3n+3}\}$ à ordonnancer dans cet ordre sur la machine et à livrer dans le même ordre vers $3n + 3$ sites $L_1, L_2, \ldots, L_{3n+3}$. Les durées

d'exécution des tâches sur la machine sont définies de la manière suivante :

$$\begin{aligned}
&p_1 = p_2 = p_3 = 0, && \\
&p_{3i+1} = p_{3i+2} = 1, && \forall i = 1 \dots n-1 \\
&p_{3i+3} = 4x_i + b_i - 2, && \forall i = 1 \dots n-1 \\
&p_{3n+1} = 4x_n + b_n + Q/2, && \\
&p_{3n+2} = p_{3n+3} = 0 &&
\end{aligned}$$

- Où x_i est défini comme suit :

$$\begin{aligned}
x_i &= (3a_i - 2b_i + 3(n-i)(a_i - b_i))/2 && \forall i = 1 \dots n \\
x_{n+1} &= 0 &&
\end{aligned} \tag{2.4}$$

- Etant donné que seuls les temps de transport $t_{j,j+1}$ et ceux entre les sites et le dépôt $t_{j,M} = t_{M,j}$ nous intéressent, ceux-ci sont définis comme suit :

$$\begin{aligned}
&t_{M,3i+1} = t_{M,3i+2} = t_{M,3i+3} = x_{i+1}, && \forall i = 0 \dots n-1 \\
&t_{3i+1,3i+2} = a_{i+1}, && \forall i = 0 \dots n-1 \\
&t_{3i+2,3i+3} = b_{i+1}, && \forall i = 0 \dots n-1 \\
&t_{3i+3,3i+4} = x_{i+1} + x_{i+2} && \forall i = 0 \dots n-1 \\
&t_{M,3n+1} = t_{M,3n+2} = t_{M,3n+3} = 0 && \\
&t_{3n+1,3n+2} = t_{3n+2,3n+3} = 0 &&
\end{aligned}$$

- La capacité du véhicule est $c = 2$ (i.e., le véhicule peut transporter au plus deux tâches durant une même tournée).

- Le problème consiste à déterminer s'il existe une solution telle que le total des dates de livraisons ($\sum_{j=1}^{n} D_j$) n'excède pas :

$$f^* = \sum_{i=1}^{n} (3C_{3i} + 7x_i + b_i) + C_{3n+1} + C_{3n+2} + C_{3n+3} - Q/2. \tag{2.5}$$

Pour la suite de la démonstration, une solution qui satisfait (2.5) est appelée *faisable* et l'ensemble des tâches $(J_{3i+1}, J_{3i+2}, J_{3i+3})$, $i = 0, \dots, n$, est appelé *triplet* et est noté T_{i+1}.

La démonstration se déroule selon le schéma suivant :

1. nous établissons d'abord via le lemme 2.2.2 que si une solution *faisable* existe alors celle-ci a une structure particulière appelée *triplet-orientée*,
2. nous analysons certaines propriétés de cette structure,
3. nous montrons qu'une solution *triplet-orientée* avec une valeur f^* existe si et seulement si la réponse au problème MEOP est positive.

Lemme 2.2.2 *Si une solution faisable existe alors il en existe une satisfaisant la propriété suivante : pour tout $i = 1, \dots, n$, les tâches J_{3i} et J_{3i+1} ne sont pas dans le même lot.*

Démonstration. On suppose qu'une solution faisable existe dans laquelle, pour un certain i ($1 \leq i \leq n$), les tâches J_{3i} et J_{3i+1} appartiennent au même lot. Etant donné que $c = 2$, le véhicule ne peut pas livrer une troisième tâche dans la même tournée. Comme conséquence à cela, le véhicule doit retourner au dépôt M afin de charger d'autres tâches et commencer une nouvelle tournée. Si on dénote par τ la date de départ de la tournée qui livre les tâches J_{3i} et J_{3i+1}, alors la tâche J_{3i} est délivrée à la date $D_{3i} = \tau + t_{M,3i}$ (date d'arrivée du véhicule au site L_{3i}) et la tâche J_{3i+1} est délivrée à la date $D_{3i+1} = \tau + t_{M,3i} + t_{3i,3i+1}$. Par conséquent, nous avons $D_{3i} = \tau + x_i$, $D_{3i+1} = \tau + x_i + (x_i + x_{i+1})$ et le véhicule retourne à M à la date $\tau + 2x_i + 2x_{i+1}$. Maintenant, si on remplace ce lot par deux lots en affectant une tâche à chacun, alors on obtient que les dates de livraison de ces tâches restent inchangées ainsi que la date de retour du véhicule au dépôt. Par conséquent, il existe une solution équivalente dans laquelle les tâches J_{3i} et J_{3i+1} ne sont pas dans le même lot. □

Ainsi, nous appelons *triplet-orientée* une solution qui satisfait le Lemme 2.2.2. La raison de cette appellation vient du fait que la solution est décomposée en triplets. Plus précisément, chaque triplet est noté $T_{i+1} = (J_{3i+1}, J_{3i+2}, J_{3i+3})$, $i = 0, \ldots, n-1$. Pour chaque triplet, une conséquence du Lemme 2.2.2 est qu'il y a exactement deux lots pour la distribution et uniquement deux possibilités pour leur composition :

— soit le premier lot contient $\{J_{3i+1}, J_{3i+2}\}$ et le second contient $\{J_{3i+3}\}$,

— soit le premier contient $\{J_{3i+1}\}$ et le second $\{J_{3i+2}, J_{3i+3}\}$.

Ces deux possibilités sont appelées *option A* et *option B* respectivement (voir Fig. 2.2). Considérons dans la suite l'option B comme étant l'*option de base*.

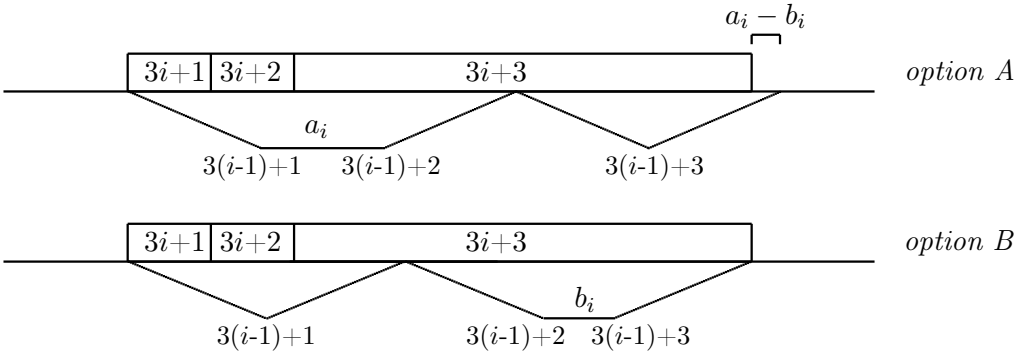


FIGURE 2.2 – Les tournées suivant les options A et B.

Taille des tournées. On désigne par taille d'une tournée le temps nécessaire au véhicule pour délivrer les tâches d'un lot. Soient M_i^A et M_i^B qui dénotent les durées nécessaires pour livrer les tâches de T_i suivant l'*option A* (et l'*option B*, respectivement) à savoir :

$$M_i^A = 4x_i + a_i \tag{2.6}$$

$$M_i^B = 4x_i + b_i \tag{2.7}$$

Etant donné que $a_i > b_i$, l'*option A* nécessite une durée plus longue pour effectuer la tournée que l'*option de base*. La différence de durées entre ces deux options (i.e., le temps supplémentaire nécessité par l'option A par rapport à l'option B) est précisément $a_i - b_i$.

Lemme 2.2.3 *Dans toute solution* triplet-orientée*, le véhicule n'est jamais en inactivité excepté potentiellement avant de charger la tâche* J_{3n+1}.

Démonstration. Pour commencer, reconsidérons d'abord le premier triplet T_1. Le véhicule part pour la tournée à la date 0 pour délivrer le lot $\{J_1\}$ (ou $\{J_1, J_2\}$) et retourne au dépôt à la date $4x_1 + b_1$ (ou $4x_1 + a_1$, respectivement). La date de fin d'exécution du triplet $T_2 = (J_4, J_5, J_6)$ est précisément égale à $C_6 = \sum_{j=1}^{6} p_j = 1 + 1 + 4x_1 + b_1 - 2 = 4x_1 + b_1$. Puisque $a_1 > b_1$, immédiatement après le retour au dépôt, le véhicule peut repartir pour livrer les tâches du triplet T_2. De la même façon, la taille des tournées (M_i^A ou M_i^B) qui livrent les tâches de T_i ne peut pas être inférieure à la durée de production du triplet T_{i+1}. Ainsi le véhicule pourra toujours repartir immédiatement pour les tâches du triplet T_{i+1}. Ce raisonnement n'est plus valide pour le dernier triplet T_{n+1} étant donnée la durée p_{3n+1} de la tâche J_{3n+1}. □

Compte tenu du Lemme 2.2.3, on peut calculer le temps total de livraison pour le scénario de base, i.e., lorsque l'option B est systématiquement choisie. D'après (2.7), on sait qu'après avoir livré les deux dernières tâches de T_i, le véhicule retourne toujours à M exactement à la date C_{3i} (voir Fig. 2.3). Par conséquent, après avoir livré les triplets $T_1, \ldots, T_n$ le véhicule retourne au dépôt à la date $C_{3n} + 4x_n + b_n$. En raison de la définition de p_{3n+1} et du fait que la production de la tâche J_{3n+1} commence à la date C_{3n}, on a $C_{3n} + 4x_n + b_n = C_{3n+1} - Q/2$. Dans ce cas, le véhicule reste en état d'inactivité au dépôt entre $C_{3n+1} - Q/2$ et C_{3n+1} et repart une fois la production de la tâche J_{3n+1} terminée. Les temps de transport vers les sites de T_{n+1} étant nuls ainsi que les durées de production des deux derniers jobs, les tâches de T_{n+1} peuvent toutes être livrées à la date $C_{3n+1} = C_{3n+2} = C_{3n+3}$. Enfin, vu que $C_1 = C_2 = C_3 = 0$, les tâches J_{3i+1}, J_{3i+2} et J_{3i+3} d'un triplet T_{i+1} sont livrées aux dates $C_{3i} + x_i$, $C_{3i} + 3x_i$, et $C_{3i} + 3x_i + b_i$, respectivement (voir Fig. 2.3).

Donc on obtient :

$$f^{BASE} = \sum_{i=1}^{n} (3C_{3i} + 7x_i + b_i) + C_{3n+1} + C_{3n+2} + C_{3n+3} \tag{2.8}$$

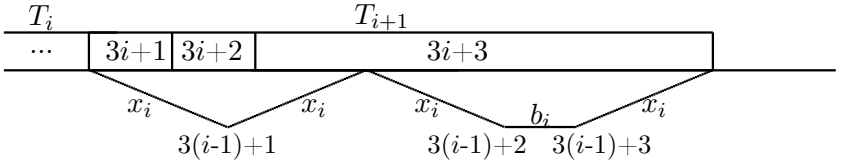


FIGURE 2.3 – La solution de base (i.e., l'option B est toujours choisie).

Contribution à la durée de livraison totale. Avant de calculer la contribution d'un certain triplet au total des dates d'arrivées, considérons une solution dans laquelle le départ du véhicule pour la livraison des trois derniers jobs J_{3n+1}, J_{3n+2} et J_{3n+3} coïncide avec leur date de fin de production, i.e., à la date $C_{3n+1} = C_{3n+2} = C_{3n+3}$ (option A et option B sont équivalentes). On appelle *régulière* une solution respectant de telles conditions.

L'expression (2.8) fait référence au scenario dans lequel l'option B est systématiquement choisie pour tous les triplets. Dans un but de clarté, on appellera par la suite *solution basique*, une solution dans laquelle nous supposons que le véhicule part livrer le triplet T_i à la date C_{3i}. A présent nous allons calculer la valeur de la fonction objectif pour une solution quelconque, i.e., une solution dans laquelle les options A et B peuvent être choisies pour tous les triplets. Tout d'abord nous allons évaluer la contribution d'un triplet T_i au total des dates d'arrivées dans une *solution basique*. Soient maintenant TDT_i^A et TDT_i^B les contributions des options A et B à la fonction objectif. On a :

$$\begin{aligned} TDT_i^A &= (C_{3i} + x_i) + (C_{3i} + x_i + a_i) + (C_{3i} + 3x_i + a_i) = 3C_{3i} + 5x_i + 2a_i \\ TDT_i^B &= (C_{3i} + x_i) + (C_{3i} + 3x_i) + (C_{3i} + 3x_i + b_i) = 3C_{3i} + 7x_i + b_i \end{aligned}$$

La différence entre les contributions selon l'option A et l'option B est comme suit :

$$\begin{aligned} TDT_i^B - TDT_i^A &= 2x_i + b_i - 2a_i \\ &= (3a_i - 2b_i + 3(n-i)(a_i - b_i)) + b_i - 2a_i \\ &= a_i - b_i + 3(n-i)(a_i - b_i) \end{aligned} \tag{2.9}$$

Sachant que $a_i > b_i$, on peut voir que la différence est positive. Cela revient à dire que le choix de l'option A au dépend de l'option B apporte un avantage en terme de total des dates de livraisons. Cependant, cette situation à première vue favorable à l'option A est atténuée par le fait que la durée de la tournée dans l'option B est plus courte (Fig. 2.4). Dans une solution *régulière*, un tel écart dans les tailles des tournées se répercute sur toutes les tâches ultérieures excepté les trois dernières J_{3n+1}, J_{3n+2} et J_{3n+3}. Pour une solution *régulière*, l'effet de choisir sur un triplet T_i l'option A par rapport à l'option B est de :

$$3(n-i)(a_i - b_i) \tag{2.10}$$

En conséquence, le *bénéfice net* en terme de fonction objectif lorsque pour un triplet T_i l'option A est choisie au dépend de l'option B est obtenu en soustrayant (2.10) à (2.9). D'après la définition de x_i (2.4), le *bénéfice net* est comme suit :

$$\begin{aligned} \text{bénéfice net}_i &= (2x_i + b_i - 2a_i) - 3(n-i)(a_i - b_i) \\ &= a_i - b_i \end{aligned} \tag{2.11}$$

Pour conclure, il s'avère que lorsque l'option A est choisie au dépend de *l'option de base*, la tournée se trouve plus grande de $(a_i - b_i)$ mais la contribution au total des dates d'arrivées plus petite de $(a_i - b_i)$ également (voir Fig. 2.5). Etant donné alors n'importe quelle solution *régulière triplet-orientée* dans laquelle les trois dernières tâches partent à leur date de fin de production, soit T_A l'ensemble des triplets pour lesquels l'option A est

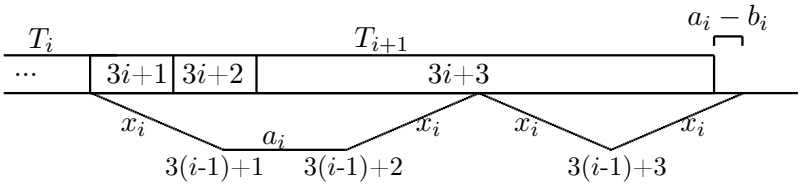


FIGURE 2.4 – T_i est le premier triplet dans lequel l'option A est choisie.

choisie. Alors, à partir de ces considérations, la valeur f de la fonction objectif est donnée par :

$$f = f^{BASE} - \sum_{i \in T_A} (a_i - b_i) \tag{2.12}$$

D'autre part, la date à laquelle le véhicule retourne au dépôt M avant le chargement des trois dernières tâches (J_{3n+1}, J_{3n+2} et J_{3n+3}) est donnée par :

$$C_{3n} + 4x_n + b_n + \sum_{i \in T_A} (a_i - b_i) \tag{2.13}$$

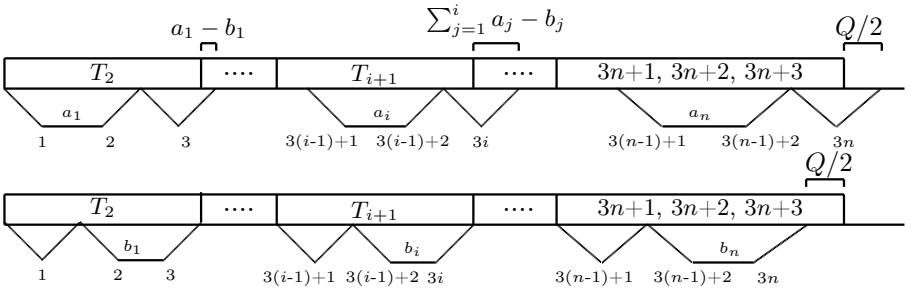


FIGURE 2.5 – Solution suivant que l'option A ou B est choisie uniquement

Dans une solution *régulière*, la tâche J_{3n+1} ainsi que les tâches J_{3n+2} et J_{3n+3} sont livrées à la date $C_{3n+1} = C_{3n} + 4x_n + b_n + Q/2$. Par conséquent, et à partir de (2.13), dans une solution *régulière* on doit avoir :

$$\sum_{i \in T_A} (a_i - b_i) \leq Q/2$$

D'un autre côté, en comparant (2.5), (2.8) il se trouve que :

$$f^* = f^{BASE} - Q/2$$

Et à partir de (2.12), une solution *régulière* est *faisable* précisément s'il existe un sous-ensemble d'indices dans T_A tel que $\sum_{i \in T_A} (a_i - b_i) = Q/2$, i.e., si et seulement si il existe une partition dans l'instance de MEOP. En conclusion de la démonstration, il reste à montrer que f^* peut être atteint uniquement par une solution *régulière*. En effet, dans une solution non *régulière*, la date de livraison des trois dernières tâches (J_{3n+1}, J_{3n+2} et J_{3n+3})

est retardée de $(\sum_{i \in T_A}(a_i - b_i) - Q/2)$. En conséquence, l'expression de f (2.12) doit être modifiée pour tenir compte d'un tel retard, à savoir :

$$f = f^{BASE} - \sum_{i \in T_A}(a_i - b_i) + 3(\sum_{i \in T_A}(a_i - b_i) - Q/2) = f^{BASE} + 2\sum_{i \in T_A}(a_i - b_i) - 3Q/2 \quad (2.14)$$

Etant donné que dans une solution non *régulière*

$$\sum_{i \in T_A}(a_i - b_i) > Q/2,$$

A partir de (2.14), on a :

$$f = f^{BASE} + 2\sum_{i \in T_A}(a_i - b_i) - 3Q/2 > f^{BASE} + Q - 3Q/2 = f^{BASE} - Q/2$$

Par conséquent, la solution ne peut pas être *faisable.* □

2.2.1 Algorithme pseudo-polynomial pour $1 \to D|k, v = 1, c,$ ***fixed-seq***$|\sum f_j D_j$

A moins que P=NP, le théorème 2.2.1 implique qu'il n'existe pas d'algorithme polynomial pour résoudre le problème $1 \to D|k, v = 1, c,$ *fixed-seq*$|\sum f_j D_j$. En conséquence à cela, un algorithme polynomial ne peut être trouvé pour des fonctions plus générales telles que $Z = \sum_j f_j(D_j)$. Dans ce qui suit, nous montrons que $1 \to D|k, v = 1, c,$ *fixed-seq*$|\sum f_j D_j$ peut être résolu en temps pseudo-polynomial ce qui permet de statuer sur la complexité de $1 \to D|k, v = 1, c,$ *fixed-seq*$|\sum f_j D_j$.

Etant donné que la séquence de production σ est prédéfinie, les dates de fin de production des tâches C_j qui représentent aussi les dates auxquelles les tâches sont prêtes pour la livraison sont connues. Sans perte de généralité, nous supposons qu'à l'instant 0 que le véhicule se trouve au dépôt. On commence par présenter les notations utilisées avant de passer à la fonction récursive.

— $\{i, j\}$: un lot contenant les tâches $J_i, \ldots, J_j$
— $M(i, j)$: la durée de la tournée transportant le lot $\{i, j\}$
— $K(i, j, t)$: la contribution à l'objectif du lot $\{i, j\}$ si le véhicule part à l'instant t

On dénote aussi par $F(i, j, t)$ la valeur de la solution optimale du problème restreint aux j premières tâches dans lequel la dernière tournée part à l'instant t et transporte le lot $\{i, j\}$. Alors $F(i, j, t)$ peut être calculé au moyen d'une formule récursive simple. Si dans l'avant dernière tournée effectuée, le véhicule transporte le lot $\{p, i-1\}$ et part à la date s, alors on a :

$$F(i, j, t) = F(p, i-1, s) + K(i, j, t)$$

A noter que si le véhicule part pour livrer le lot $\{p, i-1\}$ à la date s, celui-ci doit retourner au dépôt au plus tard à la date t. La contrainte suivante prévient du cas où le véhicule rentre après la date t.

$$C_{i-1} \leq s \leq t - M(p, i-1)$$

En conclusion, le problème est résolu au moyen de la formule récursive suivante :

$$F(i, j, t) = \min_{\substack{\max(i-c,1) \leq p \leq i-1 \\ C_{i-1} \leq s \leq t - M(p,i-1)}} \{F(p, i-1, s)\} + K(i, j, t) \quad (2.15)$$

Soit T une borne supérieure de la date de départ du véhicule pour livrer le dernier lot. Du moment que l'inégalité triangulaire est respectée, celle-ci est donnée à titre d'exemple par :

$$T = \max\left(\max_{1\leq i\leq n-1}\{C_i + 2\sum_{h=i}^{n-1} t_{hM}\}, C_n\right)$$

Enfin, la valeur de la solution optimale z^* est donnée par :

$$z^* = \min_{n-c+1\leq i\leq n, C_n\leq t\leq T}\big(F(i,n,t)\big)$$

On impose par ailleurs une limite pour éviter la situation où j est inférieur à i (2.16) et une solution initiale est donnée à l'algorithme (2.17) :

$$F(i,j,t) = +\infty \text{ for all } j < i \tag{2.16}$$

$$F(1,j,t) = K(1,j,t) \text{ for all } j,t \tag{2.17}$$

Passons maintenant à la complexité de l'algorithme en commençant par les complexités des calculs de $M(i,j)$ et $K(i,j,t)$. Ces deux fonctions peuvent être calculées facilement en ajoutant la contribution de chaque nouvelle tâche d'un lot à la durée totale de la tournée dans le cas de $M(i,j)$ ou à la fonction objectif dans celui de $K(i,j,t)$. Plus précisément, le temps de transport d_h entre le dépôt et le site d'une tâche J_h est donné par : $d_h = \begin{cases} d_{h-1} + t_{h-1,h}, & \text{if } i < h \leq j \\ t_{M,h}, & \text{if } h = i \text{ (dans le cas où } J_h \text{ est la première du lot)} \end{cases}$
Alors $M(i,j) = d_j + t_{j,M}$ et $M(i,j+1) = M(i,j) - t_{j,M} + t_{j,j+1} + t_{j+1,M}$. Etant donné que $j \leq i + c - 1$, Ceci signifie que tous les $M(i,j)$ peuvent être calculés en $O(nc)$ vu que $j \leq i + c - 1$. De la même façon, si le véhicule part à la date t pour livrer un lot $\{i,j\}$, alors la contribution à la fonction objectif d'une tâche J_h est donnée par :

$$f_h(t + d_h), \forall i \leq h \leq j$$

avec $\sum_{h=i}^{j} f_h(t+d_h)$ qui est égal à $K(i,j,t)$. A nouveau, $d_{j+1} = d_j + t_{j,j+1}$ et $K(i,j+1,t) = K(i,j,t) + f_{j+1}(d_{j+1})$. Ainsi, le calcul de toutes valeurs de $K(i,j,t)$ se fait en $O(ncT)$.

Une fois les valeurs de $M(i,j)$ et $K(i,j,t)$ connues, nous passons à la complexité de la formule (2.15) pour tous les triplets (i,j,t). Un tel calcul nécessite à chaque étape la comparaison entre $O(cT)$ valeurs. Etant donné qu'il y a $O(ncT)$ triplets possibles, le calcul de $F(i,j,t)$ se fait en $O(nc^2T^2)$ et domine clairement celui des autres phases. Ainsi, la preuve du résultat suivant est terminée.

Théorème 2.2.4 *Le problème* $1 \to D|k, v = 1, c,$ fixed-seq$|\sum f_j D_j$ *peut être résolu en* $O(nc^2T^2)$.

2.3 Autres résultats de complexité

Nous abordons dans cette section deux cas particuliers du problème $1 \to D|k, v = 1, c,$ *fixed-seq*$|\sum f_j D_j$. Dans le premier cas noté $1 \to D|k < n, v = 1, c,$ *fixed-seq*$|\sum D_j$ (section 2.3.1), on considère que le nombre de sites (clients) n'est pas forcement égal au nombre de tâches. Ensuite nous nous concentrons dans la section 2.3.2 au problème noté $1 \to D|k, v = 1, c,$ *fixed-seq, split-deliv*$|\sum f_j D_j$. Dans ce deuxième cas, on considère que la préemption est autorisée .i.e., le cas où une tâche est transportée en plusieurs parties.

2.3.1 Problème $1 \rightarrow D|k, v = 1, c,$ *fixed-seq*$|\sum D_j$ avec un nombre quelconque de sites

Le site de livraison d'une tâche J_j est noté ℓ_j et nous considérons qu'il existe k sites successifs. Ce qui revient à dire que deux tâches successives J_j et J_{j+1} sont soit à livrer au même client et dans ce cas $\ell_{j+1} = \ell_j$, soit à deux clients successifs et alors on a $\ell_{j+1} = \ell_j + 1$. A noter que pour un nombre de sites $k = n$, ce problème est équivalent a $1 \rightarrow D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$ qui est NP-difficile. A présent, nous montrons que pour une fonction objectif de type somme des dates d'arrivées, le problème demeure NP-difficile même dans le cas où le nombre de sites (clients) est strictement inférieur à n. Pour cela, nous utilisons un raisonnement semblable à celui employé pour caractériser la complexité de $1 \rightarrow D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$.

Théorème 2.3.1 *Le problème* $1 \rightarrow D|k, v = 1, c,$ fixed-seq$|\sum D_j$ *est NP-difficile.*

Démonstration. Le problème est dans NP. Etant donnée une instance de MEOP, nous construisons une instance de P3 de la façon suivante :

- Un ensemble de $3(n+1)l$ tâches $J_1, J_2, \ldots, J_{3(n+1)l}$ ordonnancées dans cet ordre sur la machine.

- $3(n+1)$ sites $\ell_1, \ell_2, \ldots, \ell_n$.

- A chaque site de livraison correspondent l tâches, i.e., l tâches doivent être livrées vers chaque site.

- On note J^k la séquence de tâches $(J_1^k, J_2^k, \ldots, J_l^k)$ qui correspond aux tâches à destination du site k. Ainsi, une tâche J_j^k correspond à la tâche $J_{l(k-1)+j}$ et est produite (en l'occurrence livrée) en position $l(k-1)+j$.

Puisque le raisonnement est semblable à celui employé pour démontrer la complexité de $1 \rightarrow D|v = 1, c,$ *fixed-seq*$|\sum D_j$, nous détaillons uniquement l'extension de la réduction polynomiale au cas où le nombre de sites est limité.

- Les durées d'exécution p_j^k des tâches sur la machine sont définies de la manière suivante :

$$
\begin{aligned}
&p_j^1 = p_j^2 = p_j^3 = 0, && \forall j = 1, \ldots, l \\
&p_1^{3i+1} = p_1^{3i+2} = 1, && \forall i = 1, ..., n-1 \\
&p_j^{3i+1} = p_j^{3i+2} = p_j^{3i+3} = 0, && \forall i = 1, ..., n-1, \forall j = 2, ..., l \\
&p_1^{3i+3} = 4x_i + b_i - 2, && \forall i = 1, ..., n-1 \\
&p_1^{3n+1} = 4x_n + b_n + Q/2, && \\
&p_j^{3n+1} = 0, && \forall j = 2, \ldots, l \\
&p_j^{3n+2} = p_j^{3n+3} = 0, && \forall j = 1, \ldots, l
\end{aligned}
$$

- x_i est défini comme suit :

$$
\begin{aligned}
x_i = \big((a_i - b_i)/l - b_i + 2a_i + 3(n-i)(a_i - b_i)\big)/2, \qquad & \forall i = 1, \ldots, n \\
x_{n+1} = 0 &
\end{aligned}
$$

A noter que $\sum_{j=1}^{l}(p_j^{3i+1} + p_j^{3i+2} + p_j^{3i+3}) = 4x_i + b_i$, $\forall i = 1, \ldots, n-1$.

Les temps de transport sont similaires à ceux utilisés dans la section 2.2 en considérant une tâche J_i comme l'ensemble de tâches J^i. De la même manière, un *triplet* T_{i+1}, $i = 0, \ldots, n$, correspond à l'ensemble de séquences de $3l$ tâches $(J^{3i+1}, J^{3i+2}, J^{3i+3})$.

- La capacité du véhicule est $c = 2l$.

- Enfin le problème consiste à déterminer s'il existe une solution telle que le total des dates de livraisons $(\sum_{j=1}^{3(n+1)l} D_j)$ n'excède pas :

$$f^* = l\sum_{i=1}^{n}(3C_{3li} + 7x_i + b_i) + \sum_{j=1}^{3l} C_{3n+j} - Q/2. \tag{2.18}$$

Une solution qui satisfait (2.18) est appelé "*faisable*".

Lemme 2.3.2 *Si une solution faisable existe alors elle satisfait la propriété suivante : pour tout k, $k = 1, \ldots, 3(n+1)$, les tâches J_j^k avec $j = 1, \ldots, l$ sont dans le même lot.*

Démonstration. On suppose qu'une solution faisable existe. Soit k le premier site pour lequel il existe deux tâches J_i^k et J_{i+1}^k $(i < l)$ qui ne sont pas livrées durant la même tournée. La production des tâches du *triplet* $T_{\lceil k/3 \rceil}$ – qui contient les l tâches de J^k – est terminée lorsque le véhicule part pour livrer les i premières tâches de J^k. On distingue deux cas : premièrement, nous supposons que la capacité du véhicule permet de prendre la tâche J_{i+1}^k. Soit τ la date de livraison de la tâche J_i^k. La tâche J_{i+1}^k ne peut pas être livrée avant la date $\tau + 2x_{\lceil k/3 \rceil}$. Ainsi, on peut voir que le total des dates de livraison ne peut croître si la tâche J_{i+1}^k est livrée avec la tâche J_i^k. Par conséquent, J_{i+1}^k peut être livrée au même temps que J_i^k et la solution reste *faisable*. Deuxièmement, nous supposons que la capacité du véhicule ne permet pas de prendre la tâche J_{i+1}^k. Etant donné que le véhicule n'a transporté que des multiples de l tâches (soit l ou $2l$ tâches) vers les sites $1, \ldots, k-1$ à chacune des tournées, cette situation est impossible. Ce qui implique qu'à chaque tournée, le véhicule peut soit prendre toutes les tâches de l'ensemble J^k, soit aucune tâche de J^k. En appliquant ce raisonnement à chaque fois, on se retrouve avec une solution dans laquelle les tâches d'un même site sont livrées ensemble. Ainsi, une solution *faisable* satisfaisant le Lemme est obtenue. □

D'après le Lemme 2.3.2, nous pouvons considérer les tâches de J^k, $k = 1, \ldots, 3(n+1)$ comme étant une seule tâche et ainsi le reste la démonstration devient similaire à celle du problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$. □

2.3.2 Problème $1 \to D|k, v = 1, c,$ *fixed-seq,split-deliv*$|\sum D_j$ avec préemption dans la livraison

Nous considérons que la préemption est autorisée pendant la livraison, i.e., une tâche peut être livrée en plusieurs partie. Le véhicule peut transporter une partie de la tâche et retourner au dépôt afin de transporter le reste. On peux remarquer que ce problème est équivalent au problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$|\sum D_j$ montré NP-difficile.

Observation 2.3.3 *Le problème* $1 \to D|k = n, v = 1, c,$ fixed-seq,split-deliv$|\sum D_j$ *avec préemption dans la livraison est NP-difficile.*

Démonstration. Considérons que la préemption est autorisée sur l'instance utilisée pour la preuve de complexité de $1 \to D|v = 1, c,$ *fixed-seq*$|\sum D_j$. On peut voir qu'à chaque fois que le véhicule retourne au dépôt, le nombre de tâche déjà produites et prêtes pour la livraison est supérieur à la capacité c. Par conséquent, transporter une tâche en plusieurs voyages successifs n'a aucun intérêt et augmente toujours la valeur de la fonction objectif. □

Ce résultat peut être étendu au cas où le nombre de sites est fixé à $k < n$.

2.4 Cas polynomiaux

Dans cette section, nous examinons deux cas particuliers. D'abord, nous traitons le cas où les temps de transport entre des sites successifs sont constants, i.e., pour tout j, $t_{j-1,j} = t_{j,j+1} = t$ et pour tout i, $t_{i,M} = t_{M,i} = t$. Dans le second cas particulier, nous considérons que le nombre de sites est borné et que les commandes destinées à un même site se suivent dans la séquence d'ordonnancement (ce problème est montré NP-difficile pour un nombre de sites quelconque dans la section 2.3.1). Pour chacun des cas, des propriétés ainsi qu'un algorithme de programmation dynamique sont proposés.

2.4.1 Temps de transport constants

Nous abordons dans cette partie le cas particulier dans lequel les sites sont considérés équidistants. Plus précisément, nous traitons les problèmes suivants :

- problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$, t_{ij} = t|\sum D_j$, qui correspond au problème $P1$ avec des temps de transport constants,
- problème $1 \to D|k = n, v = 1, c, t_{ij} = t|\sum D_j$, qui est une extension du problème $1 \to D|k = n, v = 1, c,$ *fixed-seq*$, t_{ij} = t|\sum D_j$ au cas où la séquence n'est pas fixée.

Un algorithme polynomial est proposé pour chacun de ces deux problèmes.

2.4.1.1 Problème $1 \to D|k = n, v = 1, c,$ ***fixed-seq***$, t_{ij} = t|\sum D_j$ avec des temps de transport constants

Dans ce qui suit, nous examinons le problème de minimisation d'une fonction générale des temps de livraison sur un problème de production et de livraison avec une séquence prédéfinie et des sites équidistants. Nous commençons par analyser quelques propriétés de la solution optimale avant de passer à l'algorithme.

Clairement, chaque fois le véhicule rentre au dépôt, il peut repartir immédiatement avec un nouveau lot constitué de tâches déjà produites, ou il peut attendre la fin de production d'autres tâches afin de les livrer (voir Fig. 2.6)..

Suivant Li et al.(2005), nous appelons NSS (*Non Stop Shipment*) une séquence consécutive de tournées durant laquelle le véhicule n'attend jamais au dépôt (voir Fig. 2.7), précédée par un temps d'arrêt du véhicule au dépôt. On dénote par $NSS[i, n_i, j]$ une NSS

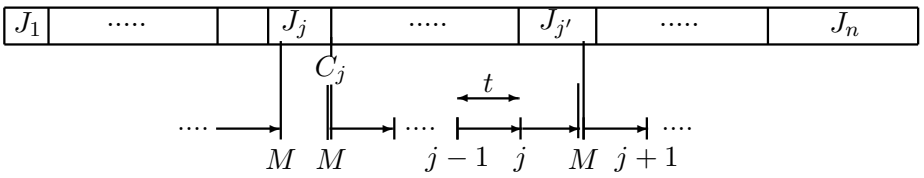


FIGURE 2.6 – Départ d'un nouveau tour

qui commence à la date de fin de production C_i de la tâche J_i et finit avant C_j, dans lequel le premier lot contient n_i tâches $\{J_{i-n_i+1}, \ldots, J_i\}$ et où J_i est la dernière tâche à être livrée durant la première tournée.

Supposons que le véhicule part pour une tournée à une certaine date τ. Soit $\mathcal{J}$ l'ensemble des tâches complétées avant τ (ou à τ) et pas encore livrées. Une tournée qui commence à τ est appelé *maximale* lorsque (i) le lot contient c jobs de $\mathcal{J}$, ou (ii) il contient tous les jobs de $\mathcal{J}$. La proposition suivante donne une caractéristique clé d'une solution optimale.

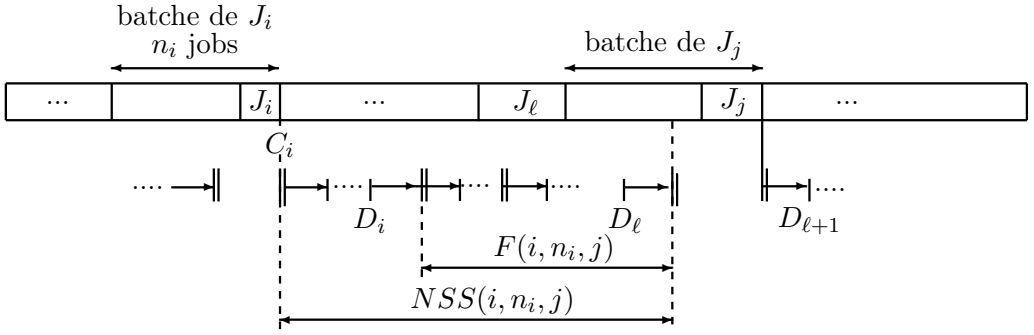


FIGURE 2.7 – Illustration d'une NSS

Proposition 2.4.1 *Il existe une solution optimale dans laquelle toutes les tournées sont maximales.*

Démonstration. On dénote par R_q la $q^{\text{ème}}$ tournée qui commence à la date τ. Soit $J_i, \ldots, J_j$ les tâches dont la production finit au plus tard à τ et pas encore livrées. La tournée R_q est maximale si elle contient $\min\{c, j-i+1\}$ tâches. On suppose que R_q n'est pas maximale, i.e. $|R_q| = k-i+1 < \min\{c, j-i+1\}$ où la dernière tâche livrée par R_q est la tâche J_k, $k \leq j-1$ (voir Fig. 2.8(a)). Cela signifie que la tâche J_{k+1} est livrée dans la tournée R_{q+1} alors qu'elle pouvait être livrée par R_q. Le fait de livrer la tâche J_{k+1} dans la tournée R_q a pour conséquence de réduire de t la date de livraison de J_{k+1} sans aucun effet sur *toutes* les autres tâches (voir Fig. 2.8(b)). Par conséquent, la nouvelle solution est meilleure que la précédente. Ce résonnement peut être répété pour toutes les tâches $k+2, \ldots, \min\{c, j-i+1\}$ et ainsi une tournée maximale est obtenue.

A partir de là, nous pouvons voir qu'une solution du problème est composée de NSS successives. Cependant, le véhicule peut avoir à attendre la fin de production de tâches supplémentaires avant de partir pour une nouvelle tournée. Cette situation est illustrée dans l'exemple suivant :

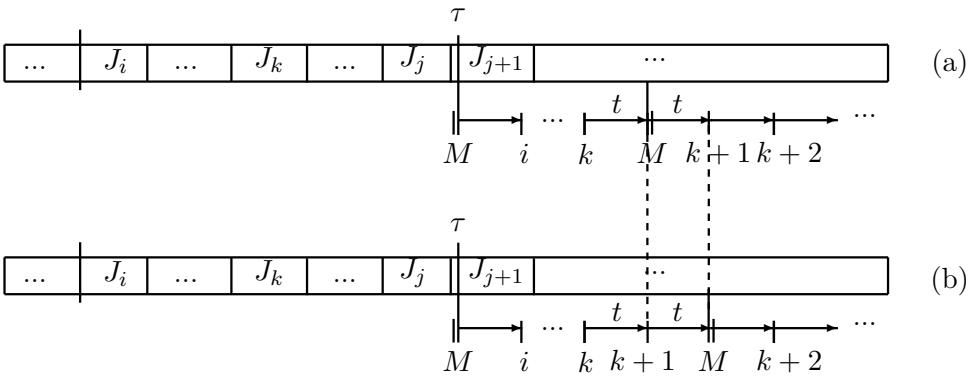


FIGURE 2.8 – Tournées maximales

Exemple : On considère une instance avec $n = 4$ tâches, les durée d'exécution sur la machine sont égales à $p = (1, 1, 10, 6)$, le temps de transport $t = 5$ et la capacité du véhicule $c = 2$. Nous allons voir les deux cas possibles dans lesquels le véhicule n'effectue aucun temps d'arrêt au dépôt (hormis pour attendre la production du premier lot). Dans le premier cas, le véhicule part à la date $C_1 = 1$ et la solution retournée induit un total des dates d'arrivées de 79. Dans le second cas, le véhicule part à $C_2 = 2$ et le total des dates d'arrivées est cette fois de 73. Cependant, la solution optimale consiste à livrer les tâches $\{J_1, J_2\}$ ensemble et au retour du véhicule au dépôt, il attend la fin de production de J_4 afin de livrer les tâches $\{J_3, J_4\}$ dans la même tournée. La solution optimale donne un total des dates de livraisons égal à 70. Les trois solutions sont illustrées dans la figure 2.9.

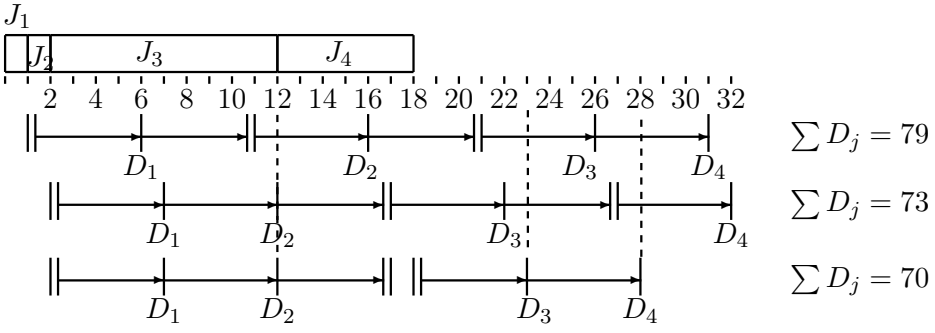


FIGURE 2.9 – Illustration de l'exemple

A partir des propriétés que nous venons de citer, nous proposons un algorithme polynomiale pour construire les différentes NSS. On note $F(i, n_i, j)$ la contribution à la fonction objectif des tâches $J_{i+1}, \ldots, J_\ell$ $(j - c \leq \ell < j)$ délivrées de la façon suivante : un NSS dénoté $NSS[i, n_i, j]$ part exactement à la date C_i, dont la première tournée délivre les tâches $\{J_{i-n_i+1}, \ldots, J_i\}$, ensuite la tâches J_{i+1} jusqu'à J_ℓ sont délivrées par des tournées maximales consécutives où, J_ℓ est la dernière tâche pouvant être livrée par le $NSS[i, n_i, j]$. On note $\nu_{i,n_i,j}$ le nombre de tâches non livrées, i.e., les tâches $J_{\ell+1}$ jusqu'à J_j et ainsi $\nu_{i,n_i,j} = j - \ell$. A la date C_j, un nouveau NSS commence dans lequel la première tour-

née est constituée des $\nu_{i,n_i,j}$ tâches pas encore livrées $J_{\ell+1}, \ldots, J_j$ (voir Fig. 2.7). On note $B_{i,j}$ le nombre de tours dans $NSS[i, n_i, j]$. Le nombre de tâches livrées par la $k^{\text{ème}}$ tournée ($1 \leq k \leq B_{i,j}$) est noté n'_k avec $n'_1 = n_i$. Étant donné que la première tournée d'un $NSS[i, n_i, j]$ commence à C_i et transporte n_i tâches alors la date de départ de la deuxième tournée est égale $C_i + t(n_i + 1)$. Ainsi, la $k^{\text{ème}}$ tournée part à la date $C_i + t\sum_{k'=1}^{k-1}(n'_{k'} + 1)$.

Il s'ensuit que

$$\begin{aligned} F(i, n_i, j) &= \sum_{k=2}^{B_{i,j}} \sum_{r=1}^{n'_k} \left(C_i + t \sum_{k'=1}^{k-1} (n'_{k'} + 1) + rt \right) \\ &= \sum_{k=2}^{B_{i,j}} \left(n'_k \big(C_i + t \sum_{k'=1}^{k-1} (n'_{k'} + 1) \big) + t \frac{n'_k(n'_k + 1)}{2} \right) \end{aligned} \tag{2.19}$$

Lorsque le nombre de tâches n_i livrées durant la première tournée ainsi que la date de départ C_i de cette première tournée sont connus, d'après la proposition 2.4.1, on peut construire en temps polynomial $NSS[i, n_i, j]$ pour tout $j > i$. Ceci implique que $F(i, n_i, j)$ peut être calculé en temps polynomial.

Soit n_0 le nombre de tâches livrées durant la première tournée. On définit $F(0, n_0, j)$ ($\forall j$, $1 \leq j \leq c$) comme étant la contribution à la fonction objectif des n_0 premières tâches $J_1, \ldots, J_j$. On a alors :

$$\begin{aligned} F(0, n_0, j) &= \sum_{l=1}^{j} (C_j + lt), \ \forall j, 1 \leq j \leq c, \text{ si } n_0 = j \\ &= \infty, \quad \text{sinon} \end{aligned}$$

Deux cas particuliers peuvent être identifiés, où il ne peut pas exister de NSS entre deux tâches J_i et J_j :

1. $C_i + t(n_i + 1) > C_j$: dans ce cas, la tournée qui livre la tâche J_i finit après la date de fin de production de J_j.
2. $\nu_{i,n_i,j} > c$: en raison de la capacité, la tâche J_j ne peut pas être livrée durant la première tournée de la prochaine NSS qui part à la date C_j.

Dans les deux cas, on pose :

$$F(i, n_i, j) = +\infty \tag{2.20}$$

Notons que les quantités $F(i, n_i, j)$ peuvent être précalculées en $O(nc)$. En effet, en calculant les quantités $F(i, n_i, n + 1)$ pour tout $i \in 1, \ldots, n$ et $n_i \in 1, \ldots, c$, les quantités $F(i, n_i, j)$ peuvent être déduites en supprimant toutes les tournées du $NSS[i, n_i, n+1]$ qui finissent après C_j.

To do this, from the $NSS[i, n_i, n + 1]$, a new $NSS[i, n_i, j]$ is built by taking all the round trips that end before C_j

On définit maintenant $f(j, n_j)$ comme étant la valeur de la solution optimale du problème restreint aux j premières tâches $\{J_1, ..., J_j\}$ dans laquelle la tournée qui livre J_j

contient n_j tâches et part à la date C_j. On a :

$$f(j,n_j) = \min_{\substack{0\leq i<j \\ 1\leq n_i\leq c \\ \nu_{i,n_i,j}=n_j}} \left\{ f(i,n_i) + F(i,n_i,j) + \sum_{r=1}^{n_j}(C_j + rt) \right\} \tag{2.21}$$

avec

$$\begin{aligned} f(0,n_j) &= 0, \quad \text{si } n_j = 0 \\ &= \infty, \quad \text{sinon} \end{aligned}$$

Puisque dans la solution optimale, la dernière tournée ne commence pas nécessairement à la date C_n, nous introduisons une tâche fictive J_{n+1} à la dernière position avec $C_{n+1} = C_n + 2tn$ et $t_{Mn+1} = t$. Clairement, ceci implique qu'il existe une solution optimale dans laquelle la dernière tournée livre uniquement la tâche fictive J_{n+1} à la date $D_{n+1} = C_{n+1}+t$.

Enfin, la solution optimale est donnée par :

$$z^* = f(n+1,1) - D_{n+1} \tag{2.22}$$

Cet algorithme de programmation dynamique peut être exécuté en $O(c^2n^2)$. Nous pouvons ainsi énoncer le théorème suivant.

Théorème 2.4.2 *Si les temps de transport sont constants, alors le problème* $1 \to D|k = n, v = 1, c,$ fixed-seq$, t_{ij} = t|\sum D_j$ *est résolu à l'optimum en temps polynomial par un algorithme de programmation dynamique en* $O(c^2n^2)$.

2.4.1.2 Problème $1 \to D|k = n, v = 1, c, t_{ij} = t|\sum D_j$ avec des temps de transport constants

Nous considérons maintenant le cas dans lequel la séquence n'est pas fixée et où les temps de transport sont constants.

Proposition 2.4.3 *Il existe une solution optimale dans laquelle les tâches sont ordonnées par ordre croissant des durées de production sur la machine (Règle Shortest Processing Time (SPT)).*

Démonstration. On suppose d'abord que les tâches sont numérotées selon la règle SPT. Ce qui veut dire que $\sigma^{SPT} = \{J_1, J_2, \ldots, J_n\}$. Supposons maintenant que la séquence SPT n'est pas optimale, alors il existe dans la séquence optimale σ^* deux tâches J_i et J_j avec $i > j$ telle que J_i précède J_j. En utilisant une permutation deux à deux des tâches, nous montrons qu'en gardant les mêmes tournées, i.e, le même nombre de tâches et les mêmes dates de départ, σ^{SPT} est telle que $\sum D_j^{SPT} \leq \sum D_j^*$ et est ainsi optimale.

Supposons que J_i et J_j sont dans la même tournée. La permutation de ces deux tâches, menant à σ' ne change pas la somme des dates de livraison et ainsi $\sum D'_j = \sum D_j^*$. Supposons que ces tâches ne sont pas dans la même tournée. Cela signifie que la tournée qui livre J_i se termine par J_i et celle qui livre J_j commence par J_j. Après la permutation de ces

deux tâches, les dates de départ de ces deux tournées restent faisables et ainsi la qualité de la solution est inchangée. Par contre, ordonnancer J_j avant J_i peut permettre de livrer J_j plus tôt et par conséquent réduire le total des dates de livraison. Nous déduisons que σ^{SPT} est optimale. □

2.4.2 Problème $1 \to D|k = K, v = 1, c,$ ***fixed-seq***$|\sum D_j$ avec un nombre de sites fixé

Nous présentons dans cette section un algorithme polynomial pour résoudre le cas particulier dans lequel un nombre de sites est fixé et égal à K et où les commandes d'un même site se suivent dans la séquence d'ordonnancement (bien entendu, la complexité de l'algorithme est exponentielle en K). A noter que ce cas particulier à été montré NP-difficile pour un nombre de sites quelconque dans la section 2.3.1. Ainsi, le théorème 2.3.1 implique qu'il n'existe pas d'algorithme polynomial pour résoudre le problème général $P1(\sum_j D_j)$ avec K sites, sauf si $P = NP$. Cependant, pour le cas particulier dans lequel les produits sont livrés vers un unique site, Li et al [Li *et al.*, 2005] ont développé un algorithme polynomial avec une complexité en $O(n^2)$ améliorant ainsi l'algorithme en $O(n^3)$ proposé par [Lee et Chen, 2001].

De la même manière que l'algorithme en section 2.4.1.1, nous introduisons une tâche fictive J_{n+1} à la dernière position avec $C_{n+1} = C_n + 2\sum_{i=1}^{n} t_{\ell_i M}$ ainsi qu'un site ℓ_{n+1} pour J_{n+1} avec $t_{\ell_{n+1},M} = 0$. La durée de production de J_{n+1} et le temps de transport $t_{\ell_{n+1},M}$ choisis impliquent qu'il existe une solution optimale du problème dans laquelle la dernière tournée livre uniquement la tâche fictive J_{n+1}.

Dans ce qui suit, on note par $F(i, n_i, j, n_j)$ la contribution à la fonction objectif des tâches $\{J_{i+1}, \ldots, J_{j-n_j}\}$ si ces tâches sont livrées par une *NSS* (noté $NSS(i, n_i, j, n_j)$) dans laquelle :

(i) La première tournée commence après que le véhicule soit retourné au dépôt après avoir livré le lot contenant les tâches $\{J_{i-n_i+1}, \ldots, J_i\}$ en partant à la date C_i.

(ii) La dernière tournée livre en dernier la tâche J_{j-n_j} et retourne au dépôt avant la date C_j.

Comme pour l'algorithme en section 2.4.1.1, $F(i, n_i, j, n_j)$ prend la valeur $+\infty$ si $NSS(i, n_i, j, n_j)$ n'existe pas. La fonction récursive est notée $f(j, n_j)$ et retourne la solution optimale du problème restreint aux j premières tâches et où le dernier lot contient n_j tâches $\{J_{j-n_j+1}, J_{j-n_j+2}, \ldots, J_j\}$ et commence à la date de fin de production C_j de la tâche J_j. Dès lors, $f(j, n_j)$ peut être calculé par la formule récursive suivante.

$$f(j, n_j) = \min_{\substack{1 \le n_i \le c \\ n_i \le i \le j - n_j}} \{f(i, n_i) + F(i, n_i, j, n_j)\} + K(j, n_j) \tag{2.23}$$

où $K(j, n_j)$ est la contribution à la fonction objectif des tâches $J_{j-n_j+1}, \ldots, J_j$ livrées par une seule tournée qui commence à la date C_j.

$$K(j, n_j) = \sum_{r=j-n_j+1}^{j} D_r, \quad \text{et} \quad D_r = C_j + t_{M,\ell_{j-n_j+1}} + \sum_{s=\ell_{j-n_j+1}}^{\ell_{r-1}} t_{s,s+1} \tag{2.24}$$

La solution optimale est donnée par :

$$z^* = f(n+1,1) - D_{n+1} \tag{2.25}$$

Afin d'établir la complexité de l'algorithme, nous commençons par donner une propriété d'un NSS, nous proposons un algorithme d'énumération qui retourne la valeur de $F(i, n_i, j, n_j)$ et enfin nous montrons que celle-ci peut être obtenu en temps polynomial.

Propriété 2.4.4 *Lorsque le véhicule repart pour effectuer une nouvelle tournée dans un NSS, s'il existe deux tâches consécutives J_l et J_{l+1} destinées au même site et dont la production est terminée. Si le véhicule livre la tâche J_l alors le véhicule livre également la tâche J_{l+1} dans la même tournée si sa capacité du véhicule le permet.*

Cette propriété est évidente. En effet, transporter la tâche J_{l+1} avec J_l ne change pas la durée de la tournée et avance la date de livraison de J_{l+1}.

L'algorithme ci-dessous utilise cette propriété pour énumérer toutes les partitions en lots qui donnent une tournée NSS. Les notations suivantes sont utilisées. Pour tout $l \in \{i, \ldots, j - n_j\}$, on note par $N(l)$ l'ensemble des labels sur J_l. Un label $(z,t) \in N(l)$ pour $l \geq i+1$ correspond à une partition des tâches $J_{i+1}, \ldots, J_l$ en lots où z représente la contribution à l'objectif des tâches $J_{i+1}, \ldots, J_l$ et t la date de retour du véhicule après avoir livré la tâche J_l. Un label initial correspondant au premier lot constitué des tâches $\{J_{i-n_i+1}, \ldots, J_i\}$ est donné par $N(i)$. A noter que $N(i)$ contient un seul label et que celui-ci est connu, i.e., la contribution à la fonction objectif du lot $\{J_{i-n_i+1}, \ldots, J_i\}$ et la date de retour du véhicule après avoir livré ce lot sont connues. Ainsi, pour tout $J_l \in \{J_{i+1}, \ldots, J_{j-n_j-1}\}$, l'algorithme étend chaque label de $N(l)$ suivant la propriété 2.4.4.

Algorithm 1 Algorithme d'énumération

$N(i) \leftarrow (Z(i - n_i + 1, i, C_i), C_i + M(i - n_i + 1, i))$
Pour $l = i$ à $j - n_j - 1$
  Pour $(z,t) \in N(l)$
    Pour $l' \in A(l,t)$
      $t' = t + M(l+1, l')$
      $z' = z + Z(l+1, l', t)$
      $N(l') \leftarrow (z', t')$
    Fin
  Fin
Fin

On note également par $M(l+1, l')$ la durée de la tournée qui livre le lot $\{J_{l+1}, \ldots, J_{l'}\}$ et par $Z(l+1, l', t))$ sa contribution à l'objectif lorsque le véhicule commence la tournée à la date t. Enfin, $A(l,t)$ est l'ensemble des lots dominants à considérer (suivant la propriété 2.4.4) lorsque le véhicule retourne à la machine à la date t après avoir livré J_l. Le prochain lot est constitué (i) des tâches du même site que J_{l+1}, ou (ii) toutes les tâches du site de J_{l+1} et les tâches du site suivant, ... et ainsi de suite tant que les tâches finissent avant t et dans la limite de la capacité du véhicule. Etant donné que $A(l,t)$ contient des lots dans

lesquels la première tâche est J_{l+1}, un lot dans $A(l,t)$ est identifié par sa dernière tâche $J_{l'}$. A noter que le plus grand lot dans $A(l,t)$ est constitué des tâches $\{J_{l+1},\ldots,J_{l+s}\}$ où $s=\min(c,\max(s'|l+s'\leq j-n_j, C_{l+s'}\leq t\})$.

Théorème 2.4.5 *L'algorithme d'énumération retourne la valeur de $F(i,n_i,j,n_j)$ en énumérant un nombre polynomial de labels.*

La démonstration suit le schéma suivant.

1. Nous définissons une borne supérieure du nombre de labels générés dans $N(j-n_j)$ (c'est à dire le nombre de NSS).
2. Nous donnons une expression polynomiale de cette borne pour une capacité c.

Point 1. Soit $h(l,z,t)$ le nombre de labels de $N(j-n_j)$ obtenus par extension du label $(z,t)\in N(l)$. On a alors d'après l'algorithme la relation

$$h(l,z,t)=\sum_{l'\in A(l,t)} h(l',z',t'),\ \text{où } t'=t+M(l+1,l') \text{ et } z'=z+Z(l+1,l',t)$$

Soit $g(l)$, pour tout $l\in\{i,\ldots,j-n_j\}$, une borne supérieure de $h(l,z,t)$ pour tout label $(z,t)\in N(l)$. Notons que comme $h(l,z,t)=\sum_{l'\in A(l,t)} h(l',z',t')$, alors

$$\max_{\tau\geq C_{l+1}} \sum_{l'\in A(l,\tau)} g(l')$$

est aussi une borne supérieure valide de $h(l,z,t)$. On peut donc choisir une borne supérieure donnée par la formulation récurrente suivante :

$$g(l)=\begin{cases} 0 & \text{for } l=j-n_j \\ 1 & \text{for } l=j-n_j-1 \\ \max_{\tau\geq C_{l+1}} \sum_{l'\in A(l,\tau)} g(l') & \text{for } l\in\{i,\ldots,j-n_j-2\}\end{cases}$$

En effet, il existe une seule manière d'étendre un label dans $N(j-n_j-1)$ du fait que seule la tâche J_{j-n_j} est disponible. La récursion provient de la définition de $h(l,z,t)$ et une borne supérieure du nombre total de labels générés par $NSS(i,n_i,j,n_j)$ dans $N(j-n_j)$ est alors donné par $g(i)$.

Point 2.

Lemme 2.4.6 *Le nombre de combinaisons de lots possibles pour livrer les tâches $J_l,\ldots,J_{j-n_j}$ de $NSS(i,n_i,j,n_j)$ tel que $l>i$ est inférieur ou égal à $2^{\ell_{j-n_j}-\ell_l}$.*

Démonstration. A noter que $g(l-1)$ est par définition une borne supérieure valide du nombre de combinaisons de lots possibles pour livrer les tâches $J_l,\ldots,J_{j-n_j}$. Ainsi, nous allons montrer que pour tout $i<l\leq j-n_j$, l'inégalité suivante est valide.

$$g(l-1)\leq 2^{\ell_{j-n_j}-\ell_l}. \tag{2.26}$$

Nous montrons par récurrence sur l que (2.26) est vraie. Suivant l'équation (2.4.2), $g(j - n_j - 1) = 1 \leq \min(2^0, 2^1)$. Par conséquent, la relation est vraie pour $l = j - n_j$.
Nous supposons que (2.26) est vraie pour tout $q \in \{l+1, \cdots, j - n_j\}$ (i.e., $g(q-1) \leq 2^{\ell_{j-n_j} - \ell_q}$) et nous montrons qu'elle est également vraie pour $q = l$. On a :

$$\begin{aligned} g(l-1) &= \max_{\tau \geq C_{l-1}} \sum_{l' \in A(l-1,\tau)} g(l') \\ &\leq \max_{\tau \geq C_{l-1}} \sum_{l' \in A(l-1,\tau) \setminus \{j-n_j\}} 2^{\ell_{j-n_j} - \ell_{l'+1}} \text{ par récurrence, car } l' \geq l \text{ et } g(j-n_j) = 0 \end{aligned}$$

Pour obtenir le pire des cas, on suppose que τ est tel que c tâches sont disponibles à la date τ (ou que que la tâche J_{j-n_j} est disponible). On a alors :

$$g(l-1) \leq \sum_{l' \in A(l-1,\ \tau) \setminus \{j-n_j\}} 2^{\ell_{j-n_j} - \ell_{l'+1}} \tag{2.27}$$

En remplaçant $\ell_{l'+1}$ par $\ell_l + (\ell_{l'+1} - \ell_l)$ dans l'équation (2.27), on a

$$\begin{aligned} \sum_{l' \in A(l-1,\ \tau) \setminus \{j-n_j\}} 2^{\ell_{j-n_j} - \ell_{l'+1}} &= \sum_{l' \in A(l-1,\ \tau) \setminus \{j-n_j\}} 2^{\ell_{j-n_j} - \ell_l - (\ell_{l'+1} - \ell_l)} \\ &= 2^{\ell_{j-n_j} - \ell_l} \sum_{l' \in A(l-1,\ \tau) \setminus \{j-n_j\}} \frac{1}{2^{\ell_{l'+1} - \ell_l}} \end{aligned} \tag{2.28}$$

Afin de calculer la valeur de l'équation (2.28), nous commençons par considérer le cas où $2^{\ell_{l'+1} - \ell_l} = 2^0$. Dans ce cas, l'ensemble de lots dominants $A(l-1,\ \tau)$ contient exactement une seule tâche (identifiant un lot) destinée au site ℓ_l. A noter que ce cas est possible lorsque toutes les tâches disponibles à la date τ sont destinées au même site. suivant l'équation (2.28), la tâche $J_{l'} \in A(l-1,\ \tau) \setminus \{j - n_j\}$ est destinée au site ℓ_l et par conséquent la tâche $J_{l'+1}$ est destinée soit au site ℓ_l, soit au site $\ell_{l'+1}$. Dans ce cas, $2^{\ell_{l'+1} - \ell_l}$ est égale alors à 1 ou 0. Ainsi, dans tous les cas, on a.

$$\sum_{l' \in A(l-1,\ \tau) \setminus \{j-n_j\}} \frac{1}{2^{\ell_{l'+1} - \ell_l}} \leq 1, \tag{2.29}$$

Ainsi, suivant les équations (2.27)-(2.29), on a.

$$g(l-1) \leq 2^{\ell_{j-n_j} - \ell_l}.$$

Cela montre que le nombre de labels créés dans $N(j - n_j)$ (borné par $g(i)$) est inférieur ou égal à 2^{K-1}. Etant donné que pour tout $l \in \{i+1, \ldots, j-n_j\}$, $N(l)$ est borné par 2^{K-1}, le nombre de label générés par l'algorithme est borné par $n2^{K-1}$. □

Il s'ensuit que lorsque K est fixé, z^* peut être calculé par un algorithme de programmation dynamique en temps polynomial en $O(n^3 c^2 2^{K-1})$. Ainsi, ce résultat de complexité paramétrée montre que l'algorithme de programmation dynamique proposé est FPT (*fixed-parameter tractable*) pour un nombre de sites k fixé.

2.4.3 Conclusion

Ce chapitre a abordé le problème d'ordonnancement et de distribution intégré à séquence fixée. Les tâches sont produites sur une seule machine puis livrées par lots vers différents clients par un seul véhicule à capacité limitée. Nous avons considéré dans ce chapitre une variante du problème dans laquelle l'ordre d'exécution des tâches sur la machine et l'ordre de livraison sont les mêmes et prédéfinis. Ainsi, le problème consiste à regrouper les tâches en lots à livrer durant la même tournée et l'objectif est de minimiser la somme des dates de livraison. Nous avons montré que le problème est NP-difficile et un algorithme de programmation dynamique pseudo-polynomial a été proposé. Plusieurs ont également été traités : nombre de sites inférieur au nombre de tâches, préemption dans la livaison, temps de transport constants et nombre de sites fixé. Pour chacun de ces cas, une preuve de complexité ou un algorithme polynomial de programmation dynamique ont été proposés.

Le cas général dans lequel la séquence n'est pas fixée est abordé dans le chapitre 4.

Chapitre 3

Problèmes d'ordonnancement et de livraison avec dates de départ fixes

En pratique dans une chaîne logistique, la production et la livraison sont traitées séparément. Ceci est d'autant plus vrai dans le cas où le système n'est pas centralisé. Dans ce chapitre, on considère que la distribution est effectuée par une entreprise tierce disposant d'un ensemble de véhicules à capacité limitée. Les produits finis sont livrés vers un unique client par des véhicules dont les dates de départ sont fixes et connues à l'avance par le fabricant. A chaque produit est associée une durée de production sur la machine et une taille qui correspond à un espace du véhicule occupé par la tâche. Puisque la distribution ne dépend pas du fabriquant, le problème consiste à déterminer une séquence de production sur la machine et à constituer les lots de tâches à livrer ensemble afin de minimiser la somme des dates de livraison des produits. Dans ce chapitre, nous montrons que ce problème est NP-difficile au sens fort, ensuite nous proposons un algorithme de branch & price pour sa résolution exacte. Un algorithme d'approximation avec garantie de performance est aussi proposé pour le cas où les durées de production et les tailles des tâches sont proportionnelles.

3.1 Description du problème et complexité

On considère un ensemble de n tâches $J = \{J_1, J_2, \ldots, J_n\}$ à produire sur une seule machine et à livrer vers un unique client. Chaque tâche J_j, $j = 1, \ldots, n$, nécessite un temps de production p_j. La livraison est effectuée par des véhicules disposant d'une capacité c et chaque tâche J_j, $j = 1, \ldots, n$, est caractérisée par un espace occupé dans le véhicule noté s_j où $0 \leq s_j \leq c$. Les dates de départ étant connues, soit τ_i la date de départ de la $i^{\text{ème}}$ tournée. On note également par μ_i le temps écoulé entre les départs τ_{i-1} et τ_i (i.e., $\mu_i = \tau_i - \tau_{i-1}$) et par t le temps de transport entre le centre de production et le client. Un lot livré à la position k est noté B_k. Les dates de départ des véhicules étant fixes alors la date de livraison d'une tâche J_j contenue dans le lot B_k est telle que $D_j = \tau_k + t$. Enfin, l'objectif est de minimiser la somme des dates de livraison des produits et suivant la notation de [Lee et Chen, 2001], le problème est noté $1 \rightarrow D, k = 1 | v = 1, c, no\ wait | \sum D_j$. Une illustration du problème est donnée dans l'exemple suivant.

Exemple : On considère une instance avec $n = 7$ tâches et une capacité du véhicule $c = 10$. Pour chaque tâche, le temps d'exécution et l'espace occupé dans le véhicule sont donnés par le tableau suivant :

J_j	1	2	3	4	5	6	7
p_j	10	17	9	11	6	13	7
s_j	3	4	5	5	4	6	5

Le temps de transport est $t = 10$ et les départs des véhicules pour effectuer la livraison sont prévus aux dates suivantes :

τ_1	τ_2	τ_3	τ_4
18	39	60	76

La figure 3.1 montre une solution du problème avec une séquence de production $\sigma = (J_3, J_5, J_1, J_4, J_7, J_6, J_2)$ et dans laquelle le véhicule livre les tâches J_3, J_5 à la date $18 + t$, les tâches J_1, J_4 à la date $39 + t$. Au départ du véhicule à la date 60, la production des tâches J_7 et J_6 est terminée mais l'espace disponible dans le véhicule ne suffit pas pour livrer ces deux tâches dans un même lot. La tâche J_7 est alors livrée à la date $60 + t$ et les deux tâches J_6 et J_2 restantes sont livrées durant la dernière tournée à la date $76 + t$.

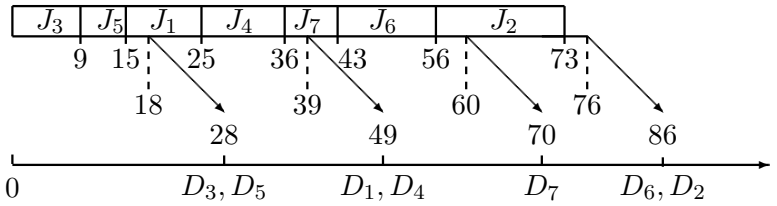


FIGURE 3.1 – Une illustration du problème $1 \to D, k = 1|v = 1, c, no\ wait| \sum D_j$

Avant de passer à la complexité et à la méthode de résolution du problème, nous reprenons quelques unes des propriétés proposées par [Chang et Lee, 2004] pour le problème $1 \to D, k = 1|v = 1, c|D_{max}$. En effet, les auteurs traitent un problème d'ordonnancement sur une machine et de livraison vers un unique client. Cependant, dans les travaux proposés par [Chang et Lee, 2004], les dates de départs des véhicules ne sont pas fixées et l'objectif à minimiser est la date de fin de livraison des tâches D_{max}. Les auteurs proposent un algorithme d'approximation amélioré successivement par [He *et al.*, 2006] et [Zhong *et al.*, 2007]. A noter que ces derniers ont proposé la meilleure approximation possible au problème. Du fait des caractéristiques de notre problème, certaines des propriétés proposées par [Chang et Lee, 2004] ne sont pas valables.

Propriétés 3.1.1 *[Chang et Lee, 2004]. Il existe une solution optimale au problème* $1 \to D, k = 1|v = 1, c, no\ wait| \sum D_j$ *telle que les propriétés suivantes sont respectées :*

- *Les tâches sont exécutées sur la machine sans interruption.*
- *Les tâches assignées au même lot sont produites consécutivement sur la machine.*
- *Les tâches assignées au même lot peuvent être exécutées suivant n'importe quel ordre.*

Par ailleurs on peut établir également facilement les propriétés suivantes.

Propriétés 3.1.2 *Il existe une solution optimale au problème* $1 \to D, k = 1 | v = 1, c, no\ wait | \sum D_j$ *telle que les propriétés suivantes sont respectées :*

- *Les lots sont produits dans l'ordre de livraison (si un lot* B_k *est produit avant le lot* B_r *alors* B_k *est livré avant* B_r*).*
- *Soit* J_j *la première tâche produite du lot* B_{k+1}, $\sum_{J_i \in B_k} s_i + s_j > c$ *ou* $\sum_{k'=1}^{k} \sum_{J_i \in B_{k'}} p_i + p_j > \tau_k$.

Démonstration. La première propriété est évidente du fait que les dates de début au plus tôt (release dates) ne sont pas prises en compte. La seconde stipule que dans une solution optimale, les lots sont complets. En effet, supposons qu'il existe un lot B_k tel que celui-ci reste réalisable si la première tâche de B_{k+1} (noté J_j) lui est rajoutée, i.e., $\sum_{J_i \in B_k} s_i + s_j \leq c$ et $\sum_{k'=1}^{k} \sum_{J_i \in B_{k'}} p_i + p_j \leq \tau_k$. La tâche J_j peut alors être affectée au lot B_k et par conséquent livrée plus tôt sans que cela n'affecte les autres lots. □

Avant de présenter la preuve de complexité du problème, nous présentons un problème fortement NP-difficile très utile pour montrer la complexité de nombreux problèmes.
`3-PARTITION`. Etant donné un ensemble de $3h$ entiers $a_1, \ldots, a_{3h}$ telles que $\sum a_i = hb$ et $b/4 < a_i < b/2$ pour tout i, est-il possible de partitionner cet ensemble d'entiers en h ensembles disjoint tels que la somme de chaque sous ensemble soit égale à b ?

Le problème de `3-PARTITION` est prouvé NP-difficile au sens fort par Garey et Johnson [Garey et Johnson, 1975]. Nous pouvons maintenant énoncer notre résultat de complexité.

Théorème 3.1.3 *Le problème* $1 \to D, k = 1 | v = 1, c = z, no\ wait | \sum_j D_j$ *is NP-difficile au sens fort.*

Démonstration. Etant donnée une instance de `3-PARTITION`, on construit une instance de notre problème de la manière suivante :

- on pose $n = 3h$ tâches et on définit la capacité $c = b$,
- le temps de transport $t > 0$ est quelconque
- pour tout $J_j \in \{J_1, \ldots, J_{3h}\}$ on pose $p_j = s_j = a_j$,
- les dates de départ sont telles que $\tau_i = ib$ pour tout $i \in 1, \ldots, 3h$,
- on pose $y = \frac{3}{2}h(h+1)b + 3ht$.

Le problème consiste alors à déterminer s'il existe une solution telle que $\sum_j D_j \leq y$. Par souci de simplicité, notre problème est noté $1 \to D$ durant cette démonstration.

De `3-PARTITION` à $1 \to D$. *S'il existe une solution à l'instance de `3-PARTITION` alors il existe pour notre problème une solution réalisable* S *telle que* $\sum_j D_j \leq y$.
Soient $H_1, H_2, \ldots, H_h$ une solution de l'instance `3-PARTITION`. On construit une solution S pour le problème $1 \to D$ telle que $\sum_j D_j = \frac{3}{2}h(h+1)b + 3ht = y$. On commence par ordonnancer les trois tâches du sous-ensemble H_1. Etant donné que $\sum_{j \in H_1} p_j = \tau_1 = b$ et $\sum_{j \in H_1} s_j = c = b$, les trois tâches constituent un lot et sont livrées au même temps à la date $b + t$. Le second départ du véhicule coïncide avec la date de fin de production du

sous-ensemble H_2. En suivant le même raisonnement, on peut voir qu'un lot B_i est prêt à la date τ_i et est livré à la date $ib+t$ et ainsi la somme des dates d'arrivées $\sum_j D_j$ est égale à $3\sum_{i=1}^{h}(ib+t) = \frac{3}{2}h(h+1)b + 3ht = y$.

De $1 \to D$ à 3-PARTITION. Inversement, on montre que *s'il existe une solution S au problème $1 \to D$ telle que $\sum_j D_j \leq y$ alors l'instance de 3-PARTITION admet une solution.* Supposons qu'une telle solution existe avec $\sum_j D_j \leq y$. Suivant l'instance générée, le nombre de lots dans S ne peut pas être inférieur à h et le nombre de tâches dans chaque lot ne peut pas excéder 3. Soit h' le nombre de lots dans S. On suppose dans un premier temps que $h' = h$ et soit S_1 la solution correspondante. Ceci implique que dans S_1, chacun des lots $b_1, b_2, \ldots, b_h$ contient exactement trois tâches et pour chaque lot $\sum_{j\in b_i} p_j = b$ et $\sum_{j\in b_i} s_j = b$. Donc, $b_1, \ldots, b_h$ définit une solution au problème de 3-PARTITION. Supposons maintenant qu'il existe une solution S_2 au problème $1 \to D$ pour laquelle $\sum_j D_j \leq y$ et $h' > h$. Soit n_1 le nombre de tâches dans $b_1, \ldots, b_h$ et n_2 le nombre de tâches dans $b_{h+1}, \ldots, b_{h'}$. On note par $\sigma_j^{S_1}$ et $\sigma_j^{S_2}$ les tâches ordonnancées à la position j dans S_1 et S_2 respectivement. Du fait que tous les lots de S_1 contiennent exactement trois tâches, on peut voir que $\sum_{j=1}^{n_1} D_{\sigma_j^{S_2}} \geq \sum_{j=1}^{n_1} D_{\sigma_j^{S_1}}$. Dans la solution S_2, les n_2 tâches restantes sont livrées après la date $hb+t$ qui représente la date de livraison de la tâche $J_{\sigma_{n_1}^{S_2}}$ et de la dernière tâche de $J_{\sigma_n^{S_1}}$ dans la solution S_1. Donc $\sum_{j=n_1+1}^{n} D_{\sigma_j^{S_2}} > \sum_{j=n_1+1}^{n} D_{\sigma_j^{S_1}}$ qui implique que $\sum_j D_j > y$ dans la solution S_2, ce qui est impossible. □

Par ailleurs nous pouvons déterminer un cas trivialement polynomial par l'observation suivante :

Observation 3.1.4 *Le problème $1 \to D, k = 1 | v = 1, c, no\ wait | \sum D_j$ peut être résolu en temps polynomial dans le cas où toutes les tâches ont la même taille. La séquence optimale consiste à ordonner les tâches suivant un ordre croissant de leurs durées d'exécution.*

3.2 Cas particulier : livraisons régulières et $s_j = f(p_j)$

Dans cette section, nous traitons un cas particulier du problème dans lequel les livraisons se font à des dates periodiques et où l'espace qu'occupe une tâche dans le véhicule est proportionnel à sa durée de production sur la machine. Soit μ le temps entre deux départs consécutifs, i.e. $\tau_{k+1} - \tau_k = \mu$ et soit f une fonction croissante telle que $\forall x, y \in \mathbb{R}$, $f(x+y) = f(x) + f(y)$. Ainsi la date de départ du lot en position k est $\tau_k = k\mu$ et la date de livraison d'une tâche J_j contenue dans le lot B_k est $D_j = k\mu + t$. Le problème considéré dans cette section tel que $s_j = f(p_j)$ à été monté NP-difficile au sens fort par une réduction au problème de 3-PARTITION (voir le théorème 3.1.3). En effet, dans la preuve de complexité du problème général $1 \to D, k = 1 | v = 1, c, no\ wait | \sum D_j$, nous avons considéré spécialement à cet effet que le temps entre deux départs était égal à b et que $s_j = p_j$ qui représente un cas particulier de la contrainte $s_j = f(p_j)$. Ce cas particulier est alors noté $1 \to D, k = 1 | v = 1, c, no\ wait, s_j = f(p_j), \tau_k = k\mu | \sum D_j$

Etant donné que ce cas particulier du problème est toujours NP-difficile au sens fort, nous proposons une méthode de résolution heuristique avec garantie de performance. Nous

commençons par présenter quelques notations ainsi que des notions nécessaires avant de passer à l'algorithme de résolution.

3.2.1 Le problème de bin-packing

On considère un ensemble de n objets $a_1, \ldots, a_n$ dont on connaît les poids p_i et un nombre infini de boites $b_1, b_2, \ldots$ de capacités identiques c. Sachant que les boites ne peuvent supporter qu'un poids maximum, le problème du bin-packing consiste à ranger tous les objets dans le minimum de boites possible. Ce problème simple en apparence est NP-difficile au sens fort.

Exemple : On considère une instance avec 8 objets et un nombre suffisant de boites de capacité $c = 8$. Les poids des objets sont donnés dans le tableau suivant :

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
2	3	1	4	5	2	3	4

Le total des poids étant égal à 24, i.e., $\sum_{i=1}^{8} p_i = 24$, une borne inférieure du nombre de boites nécessaires pour ranger tous les objets est $\frac{\sum_{i=1}^{8} p_i}{c} = 3$. Une solution nécessitant 3 boites existe dans ce cas particulier et est illustrée dans la figure suivante :

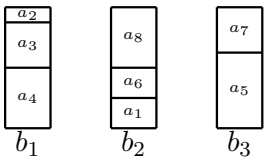


FIGURE 3.2 – Exemple de solution optimale du bin-packing

Parmi les méthodes de résolution heuristiques du bin-packing, il existe des algorithmes *gloutons* capables de résoudre le problème rapidement et de garantir une certaine performance de la solution. Parmi les algorithmes les plus étudiés, on peut citer :

Next Fit : Initialement les boites sont vides et on commence par affecter l'objet a_1 à la boite b_1. Puis itérativement en partant de j=2, si une boite b_i a une capacité résiduelle suffisante pour recevoir l'objet a_j, i.e., $c - \sum_{l \in b_i} p_l \geq p_j$ alors l'objet a_j est affecté à b_i et nous passons à l'objet a_{j+1}. Sinon, la boite b_i est fermée et a_j est affecté à une nouvelle boite b_{i+1}. Le processus est répété jusqu'à ce que tous les objets soient affectés.
A partir de l'instance utilisée dans le précédent exemple, l'heuristique *Next Fit* est appliquée. Le déroulement de cet algorithme et la solution retournée sont présentés dans la figure 3.3.

objet	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
boite	b_1	b_1	b_2	b_3	b_3	b_4	b_4	b_4

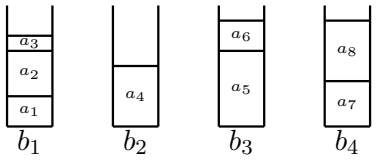


FIGURE 3.3 – Exemple d'application de l'heuristique *Next Fit*

L'algorithme *Next fit* définit une 2-approximation et est résolu avec une complexité temporelle en $O(n)$.

First Fit : A la différence de *Next fit* dans laquelle l'algorithme ne revient pas sur une boite fermée, dans l'heuristique *First Fit*, un objet a_j est affecté à la boite de plus petit indice dont la capacité restante permet de le recevoir. L'heuristique *First Fit* donne le résultat présenté dans la figure 3.4 sur l'instance de l'exemple précédent.

objet	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
boite	b_1	b_1	b_1	b_2	b_3	b_1	b_2	b_4

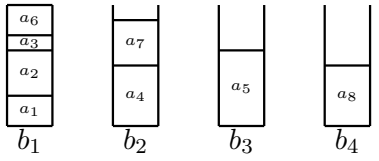


FIGURE 3.4 – Exemple d'application de l'heuristique *First Fit*

L'heuristique *First Fit* donne un ratio d'approximation absolue 1.7 [Dósa et Sgall, 2013]. i.e., si dans une solution optimale, OPT boites sont nécessaires pour ranger tous les objets, alors *First Fit* nécessite au plus $\lfloor 1.7 \cdot OPT \rfloor$ boites. Le problème est résolu avec une complexité temporelle en $O(nlog(n))$.

Best Fit : De la même manière que les heuristiques précédentes, les objets sont affectés dans l'ordre, i.e., de a_1 à a_8. A la différence de *First Fit* dans laquelle un objet a_j est affecté à la boite de plus petit indice, dans *Best Fit*, a_j est affecté à la boite la plus lourde. Le résultat suivant est obtenu en appliquant cette heuristique .

objet	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
boite	b_1	b_1	b_1	b_2	b_3	b_1	b_3	b_2

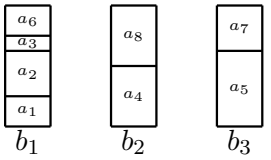


FIGURE 3.5 – Exemple d'application de l'heuristique *Best Fit*

L'heuristique *Best Fit* admet un ratio d'approximation absolue 1.7 [Dósa et Sgall, 2014] et est résolu avec complexité temporelle en $O(n^2)$.

First Fit decreasing : une amélioration naturelle de l'heuristique *First Fit* consiste à ordonner les objets par ordre décroissant des poids. Cette amélioration est apportée dans *First Fit decreasing*. La figure 3.6 donne une solution du problème donnée par l'heuristique.

objet	a_5	a_4	a_8	a_2	a_7	a_1	a_6	a_3
boite	b_1	b_2	b_2	b_1	b_3	b_3	b_3	b_3

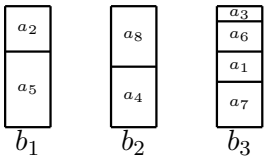


FIGURE 3.6 – Exemple d'application de l'heuristique *First Fit Decreasing*

L'heuristique *First Fit decreasing (FFD)* donne une $\frac{3}{2}$-approximation [Simchi-Levi, 1994] et est résolu avec une complexité temporelle de $O(n^2)$.

Voir [Coffman Jr *et al.*, 2013] pour plus de détails sur les algorithmes d'approximation sur le problème du bin-packing.

3.2.1.1 Le problème de vecteur-packing à d-dimensions

Ce problème est une généralisation du bin-packing à une seule dimension et est défini comme suit.

Définition 3.2.1 *Etant donné un ensemble de L vecteurs dans* $[0,1]^d$*, l'objectif est de partitionner les L vecteurs en un minimum de sous-ensembles tels que la somme des coordonnées pour chaque sous-ensemble soit au plus égale à 1.*

Exemple : On considère dans cet exemple un problème à 2 dimensions, i.e., $d = 2$ pour lequel les poids ne sont pas normalisés. A partir de l'exemple précédent du bin-packing à une dimension, supposons maintenant que chaque objet en plus de son poids a une taille et que nous disposons d'un nombre suffisant de boites $\{TB_1, TB_2, \ldots\}$ pour ranger tous les objets. Soient $c = 8$ et $c' = 10$, respectivement le poids et la taille maximum pouvant

objet	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
poids (p_i)	2	3	1	4	5	2	3	4
tailles (s_i)	2	3	4	4	4	5	2	3

être supportés par une boite. Le tableau suivant donne le poids et la taille pour chacun des objets.

Le total des poids étant égal à 24 et le total des tailles à 27, une borne inférieure du nombre de boites nécessaires pour ranger tous les objets est $\max\{\frac{\sum_{i=1}^{8} p_i}{c}, \frac{\sum_{i=1}^{8} s_i}{c'}\}$ [Spieksma, 1994]. Une solution nécessitant 3 boites est illustrée dans la figure suivante :

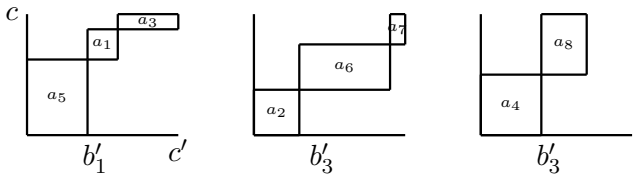


FIGURE 3.7 – Exemple de bin-packing à deux dimensions

Une heuristique admettant un ratio d'approximation absolue 2 est proposée par [Kellerer et Kotov, 2003]. Nous renvoyons à [Christensen *et al.*, 2017] pour plus de détails sur les algorithmes d'approximation sur le problème du bin-packing à plusieurs dimensions.

3.2.2 Heuristique d'approximation

L'heuristique suivante est proposée pour résoudre le problème $1 \rightarrow D, k = 1 | v = 1, c, no\ wait, s_j = f(p_j), \tau_k = k\mu | \sum D_j$.

Heuristique H

Etape 1 : Trier les tâches par ordre croissant des tailles (i.e., $s_1 \leq s_2 \leq \cdots \leq s_n$). Soit σ^H la séquence retournée.

Etape 2 : Suivant la séquence σ^H et à partir de la tâche $J_{\sigma_1^H}$ et en commençant par le lot b_1 affecter les tâches aux lots en respectant la capacité du véhicule.

Soit b_i^H un lot livré à la position i dans l'heuristique. Nous considérons b_i^H comme étant une boite à 2-dimensions telle que la première dimension concerne la capacité du véhicule avec :

$$\sum_{J_j \in b_i^H} s_j \leq c$$

et une deuxième dimension dont la taille c'_i correspond au temps écoulé entre la date de fin de production de b_{i-1}^H et la date τ_i de départ du lot b_i^H.

$$\sum_{j \in b_i^H} p_j \leq c'_i,$$

Ainsi, cette heuristique retourne des lots réalisables pour le problème $1 \rightarrow D, k = 1|v = 1, c, no\ wait, s_j = f(p_j)| \sum D_j$. L'étape 2 de cette heuristique peut également être vue comme une adaptation de l'heuristique *Next fit* sur un vecteur bin-packing à deux dimensions.
Soient S^* la solution optimale du problème et S^H la solution retournée par l'heuristique. Le total des dates d'arrivées est noté $Z(S^*)$ pour la solution optimale et $Z(S^H)$ pour la solution heuristique H. Ainsi, le théorème suivant montre que le problème est 2-approximable.

Théorème 3.2.1 $Z(S^H) < 2Z(S^*)$

Démonstration. Nous commençons par introduire quelques notations nécessaires à la preuve. On note par δ^H et δ^* le nombre total de lots dans la solution S^H et dans la solution S^* respectivement. Un lot transporté à la date μ_i est noté b_i^H dans la solution heuristique S^H et b_i^* dans la solution optimale S^*. Un lot b_q^H est appelé *complet* dans les deux cas suivants :

(A) : $\sum_{J_j \in b_q^H} s_j \leq c$ et $\sum_{J_j \in b_q^H} s_j + s_l > c$, $\forall J_l \in b_{q+1}^H$

(B) : $\sum_{j \in b_q^H} p_j \leq c'_q$ et $\sum_{J_j \in b_q^H} p_j + p_l > c'$, $\forall J_l \in b_{q+1}^H$

En d'autres termes, un lot b_q^H est complet suivant (A) si la capacité du véhicule ne peut contenir une tâche supplémentaire non transportée. Les tâches étant triées par ordre croissant des durées d'exécution, l'ajout de la première tâche du lot b_{q+1}^H rend la solution irréalisable. De la même manière, un lot b_q^H est complet suivant (B) si l'ajout d'une tâche supplémentaire fait que la date de fin de production du lot est supérieure à τ_i. A noter que d'après les propriétés (3.1.2), tous les lots sont complets selon (A) ou (B) sauf éventuellement le dernier lot. Les lemmes suivants sont utilisés dans la démonstration du théorème.

Lemme 3.2.2 *Soient $J_{\sigma_j^H}$ et $J_{\sigma_j^*}$ les tâches produites à la position j qui se trouvent dans les lots b_k^H et b_i^* dans la solution S^H et S^* respectivement. On a alors $k < 2i$.*

Démonstration. Nous distinguons deux cas. Dans le premier, nous considérons que les lots $\{b_1^H, \ldots, b_{k-1}^H\}$ qui précèdent le lot b_k^H sont complets suivant (A). Ensuite, nous examinerons le cas où il existe $q \in \{1, \ldots, k-1\}$ tel que le lot b_q^H soit complet uniquement selon (B) et pour tout $m \in \{q+1, \ldots, k-1\}$, b_m^H est complet uniquement selon (A).

Cas 1 : Les lots $\{b_1^H, \ldots, b_{k-1}^H\}$ sont complets selon (A).

Soient $P^*(1, j)$ le problème restreint aux tâches $\{\sigma_1^*, \ldots, \sigma_j^*\}$ de la solution optimale S^* et $P^H(1, j)$ le problème restreint aux tâches $\{\sigma_1^H, \ldots, \sigma_j^H\}$ de la solution approchée S^H. Dans ce cas, une borne inférieure du nombre de lots pour les problèmes $P^*(1, j)$ et $P^H(1, j)$ est une solution du bin-packing à une dimension où seules les tailles des tâches sont considérées.
Soient Δ^H et Δ^* les valeurs des solutions optimales du bin-packing pour le problème $P^H(1, j)$ et $P^*(1, j)$ respectivement. Alors on a :

$$\Delta^H \leq k \quad et \quad \Delta^* \leq i$$

Etant donné que les $k-1$ premiers lots sont complets suivant (A) sur le problème restreint aux j premières tâches $P^H(1, j)$, l'étape 2 de l'heuristique H est équivalente à l'heuristique

Next fit sur le bin-packing à une dimension où seules les tailles sont considérées. De plus, les tâches étant triées par ordre croissant des tailles, les heuristiques *Next fit*, *Best fit* et *First fit* sont équivalentes sur le problème $P^H(1, j)$.

$$k \leq \frac{17}{10}\Delta^H$$

Les j tâches de $P^H(1, j)$ étant les j plus petites tâches de l'ensemble des tâches, on obtient.

$$\Delta^H \leq \Delta^*$$

Il s'ensuit que les tâches $J_{\sigma_j^H}$ et $J_{\sigma_j^*}$ qui se trouvent respectivement dans les lots b_k^H et b_i^* sont telles que.

$$k \leq \frac{17}{10}\ i \tag{3.1}$$

Cas 2 : Il existe $q \in \{1, \ldots, k-1\}$ tel que le lot b_q^H soit complet uniquement selon (B) et pour tout $m \in \{q+1, \ldots, k-1\}$, b_m^H est complet uniquement selon (A).

Dans le premier cas, nous avons traité le cas où les lots qui précédent la tâche en position j étaient complets selon (A). A présent, nous considérons qu'il existe dans S^H un seul lot noté b_q^H dans $\{b_1^H, \ldots, b_{k-1}^H\}$ tel que b_q^H soit complet uniquement selon (B) et nous montrons que l'heuristique H retourne une solution dans laquelle la tâche produite en position j est contenue dans un lot b_k^H telle que $k < 2i$ où, b_i^* est le lot qui contient la tâche produite en position j dans la solution optimale. Les notations utilisées pour la démonstration sont les suivantes.

- $|b_u^H|$ et $|b_u^*|$: le nombre de tâches contenues dans les lots b_u^H et b_u^* respectivement
- $N_u^H = \sum_{h=1}^{u} |b_h^H|$: le nombre de tâches contenues dans les lots $b_1^H, \ldots, b_u^H$
- $N_u^* = \sum_{h=1}^{u} |b_h^*|$: le nombre de tâches contenues dans les lots $b_1^*, \ldots, b_u^*$
- $P_u^H = \sum_{h \in b_u^H} p_h$: la durée de production du lot b_u^H
- $P_u^* = \sum_{h \in b_u^*} p_h$: la durée de production du lot b_u^H
- $S_u^H = \sum_{h \in b_u^H} s_h$: la taille du lot b_u^H
- $S_u^* = \sum_{h \in b_u^*} s_h$: la taille du lot b_u^H

Le lot b_q^H étant complet uniquement selon (B), l'ajout d'une tâche supplémentaire ferait que la date de fin de production du lot b_q^H soit supérieure à la date de départ τ_q. Par ailleurs le lot est complet selon (B) et non selon (A), on obtient donc :

$$\sum_{u=1}^{q} P_u^H \leq \tau_q \tag{3.2}$$

$$\sum_{u=1}^{q} P_u^H + p_{\sigma_v^H} > \tau_q, \quad v = N_q^H + 1 \tag{3.3}$$

$$S_q^H + s_{\sigma_v^H} \leq c, \quad v = N_q^H + 1 \tag{3.4}$$

Les tâches de la solution S^H étant triées par ordre croissant de leurs durées de production et leurs tailles, la somme des durées de production ainsi que la somme des tailles des tâches

jusqu'à une position donnée est comme suit.

$$\sum_{v=1}^{w} p_{\sigma_v^H} \leq \sum_{v=1}^{w} p_{\sigma_v^*},\ \forall w \in 1,\ldots,n \quad \Rightarrow \quad \sum_{v=1}^{j} p_{\sigma_v^H} \leq \sum_{v=1}^{j} p_{\sigma_v^*} \tag{3.5}$$

$$\sum_{v=1}^{w} s_{\sigma_v^H} \leq \sum_{v=1}^{w} s_{\sigma_v^*},\ \forall w \in 1,\ldots,n \quad \Rightarrow \quad \sum_{v=1}^{j} s_{\sigma_v^H} \leq \sum_{v=1}^{j} s_{\sigma_v^*} \tag{3.6}$$

Il existe un entier $l \in \{1,\ldots,k-1\}$ et un entier $r \in \{1,\ldots,i-1\}$ tels que $k = q + l$ et $i = q + r$. A noter que $i > q$ car la production des j premières tâches dans la solution optimale finit après le $q^{\text{ème}}$ départ du véhicule. Ainsi, les tâches $J_{\sigma_j^H}$ et $J_{\sigma_j^*}$ sont dans les lots b_{q+l}^H et b_{q+r}^* respectivement. D'après les équations (3.5) et (3.6), les sommes des durées de productions et des tailles jusqu'aux tâches $J_{\sigma_j^H}$ et $J_{\sigma_j^*}$ dans les solutions S^H et S^*, respectivement, sont comme suit.

$$\begin{aligned}
&\sum_{v=1}^{N_q^H} p_{\sigma_v^H} + \sum_{v=N_q^H+1}^{j} p_{\sigma_v^H} \leq \sum_{v=1}^{N_q^*} p_{\sigma_v^*} + \sum_{v=N_q^*+1}^{j} p_{\sigma_v^*} \\
&\Leftrightarrow \sum_{u=1}^{q} P_u^H + \sum_{v=N_q^H+1}^{j} p_{\sigma_v^H} \leq \sum_{u=1}^{q} P_u^* + \sum_{v=N_q^*+1}^{j} p_{\sigma_v^*}
\end{aligned} \tag{3.7}$$

$$\begin{aligned}
&\sum_{v=1}^{N_q^H} s_{\sigma_v^H} + \sum_{v=N_q^H+1}^{j} s_{\sigma_v^H} \leq \sum_{v=1}^{N_q^*} s_{\sigma_v^*} + \sum_{v=N_q^*+1}^{j} s_{\sigma_v^*} \\
&\Leftrightarrow \sum_{u=1}^{q} S_u^H + \sum_{v=N_q^H+1}^{j} s_{\sigma_v^H} \leq \sum_{u=1}^{q} S_u^* + \sum_{v=N_q^*+1}^{j} s_{\sigma_v^*}
\end{aligned} \tag{3.8}$$

Soit $M = \sum_{u=1}^{q} P_u^* - \sum_{u=1}^{q} P_u^H$ la différence des durées totales de production des lots de 1 à q dans les solutions optimale et heuristique. Etant donné que $\sum_{u=1}^{q} P_u^* \leq \tau_q$ et que d'après (3.3), $\sum_{u=1}^{q} P_u^H + p_{\sigma_{N_q^H+1}^H} > \tau_q$, on obtient.

$$\begin{aligned}
&\sum_{u=1}^{q} P_u^* \leq \sum_{u=1}^{q} P_u^H + p_{\sigma_v^H},\quad v = N_q^H + 1 \\
&\Rightarrow \sum_{u=1}^{q} P_u^* - \sum_{u=1}^{q} P_u^H \leq p_{\sigma_v^H},\quad v = N_q^H + 1 \\
&\Rightarrow M \leq p_{\sigma_v^H},\quad v = N_q^H + 1
\end{aligned} \tag{3.9}$$

Soit maintenant $N = \sum_{u=1}^{q} S_u^* - \sum_{u=1}^{q} S_u^H$ la différence entre les tailles totales des tâches des lots de 1 à q dans les solutions optimale et heuristique. En soustrayant $\sum_{u=1}^{q} S_u^H$ à

(3.8), nous obtenons.

$$\sum_{v=N_q^H+1}^{j} s_{\sigma_v^H} \leq (\sum_{u=1}^{q} S_u^* - \sum_{u=1}^{q} S_u^H) + \sum_{v=N_q^*+1}^{j} s_{\sigma_v^*}$$

$$\Leftrightarrow \sum_{v=N_q^H+1}^{j} s_{\sigma_v^H} \leq N + \sum_{v=N_q^*+1}^{j} s_{\sigma_v^*}$$

$$\Leftrightarrow \sum_{v=N_q^H+1}^{j} s_{\sigma_v^H} - \sum_{v=N_q^*+1}^{j} s_{\sigma_v^*} \leq N \tag{3.10}$$

Etant donné que $\forall x, y \in \mathbb{R}$, $f(x+y) = f(x) + f(y)$ et que $s_v = f(p_v), \forall v \in \{1, \ldots, n\}$, l'espace N est alors fonction de la durée de production lui correspondant $\sum_{u=1}^{q} P_u^* - \sum_{u=1}^{q} P_u^H$. Ainsi, $N = f(M)$. De la même manière, nous avons d'après (3.9).

$$\begin{aligned} M \leq p_{\sigma^H_{N_q^H+1}} \quad &\Rightarrow \quad f(M) \leq f(p_{\sigma^H_{N_q^H+1}}) \\ &\Rightarrow \quad N \leq s_{\sigma_v^H}, \quad v = N_q^H + 1 \end{aligned} \tag{3.11}$$

Etant donné que les lots $b_{q+1}^H, \ldots, b_{q+l-1}^H$ sont complets uniquement selon (A), alors nous avons :

$$S_{q+u}^H + s_{\sigma_v^H} > c, \quad \forall u \in \{1, \ldots, l-1\}, v = N_{q+u}^H + 1 \tag{3.12}$$

En additionnant les $l-1$ inégalités de (3.12), nous obtenons :

$$\begin{aligned} \sum_{u=1}^{l-1} (S_{q+u}^H + s_{\sigma^H_{N^H_{q+u}+1}}) &> (l-1)c \\ \Leftrightarrow \sum_{u=1}^{l-1} S_{q+u}^H &> (l-1)c - \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} \end{aligned} \tag{3.13}$$

Dans la solution optimale S^* du problème, les tâches $J_{\sigma^*_{N_q^*+1}}, \ldots, J_{\sigma_j^*}^*$ sont affectées aux r lots $b_{q+1}^*, \ldots, b_{q+r}^*$, alors nous avons :

$$\sum_{u=1}^{h} S_{q+u}^* \leq hc, \quad \forall h = 1, \ldots, r \tag{3.14}$$

Soit λ_j^H et λ_j^* les espaces occupés par les tâches des lots b_{q+l}^H et b_{q+r}^* produites avant les tâches σ_{j+1}^H et σ_{j+1}^* respectivement, i.e., les tâches $\sigma^H_{N^H_{q+l-1}+1}, \ldots, \sigma_j^H$ et les tâches $\sigma^*_{N^*_{q+r-1}+1}, \ldots, \sigma_j^*$ respectivement.

$$\lambda_j^H = \sum_{v=N_{q+l-1}^H+1}^{j} s_{\sigma_v}^H, \quad \text{et} \quad \lambda_j^* = \sum_{v=N_{q+r-1}^*+1}^{j} s_{\sigma_v}^*$$

L'inégalité (3.10) peut s'écrire de la manière suivante.

$$\begin{aligned}(\sum_{u=1}^{l-1} S^H_{q+u}+\lambda^H_j) - (\sum_{u=1}^{r-1} S^*_{q+u} + \lambda^*_j) \leq N \\ \Rightarrow \sum_{u=1}^{l-1} S^H_{q+u} \leq \sum_{u=1}^{r-1} S^*_{q+u} + N + \lambda^*_j - \lambda^H_j \end{aligned} \tag{3.15}$$

Les inégalités (3.14) et (3.15) impliquent.

$$\sum_{u=1}^{l-1} S^H_{q+u} \leq (r-1)c + N + \lambda^*_j - \lambda^H_j \tag{3.16}$$

Les inégalités (3.13) et (3.16) impliquent.

$$(l-1)c - \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} < (r-1)c + N + \lambda^*_j - \lambda^H_j \tag{3.17}$$

La tâche en position j étant affectée aux lots b^H_{q+l} dans la solution S^H et au lot b^*_{q+r} dans la solution optimale S^*, soit $a \in \mathbb{R}^+$ tel que $q + l \geq a(q + r)$. Nous cherchons dans ce qui suit une borne supérieure de a telle que $q + l$ soit inférieur à $a(q + r)$. Nous remplaçons dans (3.17) l par $a(q + r) - q$ et nous obtenons l'inégalité suivante.

$$\begin{aligned}(a(q+r) - q - 1)c - \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} < (r-1)c + N + \lambda^*_j - \lambda^H_j \\ \Rightarrow (a-1)rc < \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} - (a-1)qc + N + \lambda^*_j - \lambda^H_j \end{aligned} \tag{3.18}$$

Etant donné que $\forall u \in 1, \ldots, l-1,\ J_{\sigma^H_{N^H_{q+u}+1}} \in b^H_{q+u+1}$ car $J_{\sigma^H_{N^H_{q+u}+1}}$ est la première tâche de b_{q+u+1}. Ainsi, nous avons.

$$\begin{aligned} & s_{\sigma^H_{N^H_{q+l-1}+1}} \leq \lambda^H_j \ \ et \ \ s_{\sigma^H_{N^H_{q+u}+1}} \leq S^H_{q+u+1},\ \forall u \in 1, \ldots, l-2 \\ & \Rightarrow \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} \leq \sum_{u=2}^{l-1} S^H_{q+u} + \lambda^H_j \\ & \Rightarrow \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} + S^H_{q+1} \leq \sum_{u=1}^{l-1} S^H_{q+u} + \lambda^H_j \end{aligned} \tag{3.19}$$

D'après les inégalités (3.15) et (3.19), nous avons.

$$\begin{aligned}
&\sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} + S^H_{q+1} \leq \sum_{u=1}^{r-1} S^*_{q+u} + N + \lambda^*_j - \lambda^H_j + \lambda^H_j \\
&\Rightarrow \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} \leq \sum_{u=1}^{r-1} S^*_{q+u} + N + \lambda^*_j - S^H_{q+1} \\
&\Rightarrow \sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}} \leq (r-1)c + N + \lambda^*_j - S^H_{q+1} \qquad (3.20)
\end{aligned}$$

En utilisant (3.20) pour remplacer $\sum_{u=1}^{l-1} s_{\sigma^H_{N^H_{q+u}+1}}$ dans (3.18), nous obtenons.

$$\begin{aligned}
&(a-1)rc < (r-1)c + N + \lambda^*_j - S^H_{q+1} - (a-1)qc + N + \lambda^*_j - \lambda^H_j \\
&\Rightarrow arc + aqc < 2rc - c + 2N + 2\lambda^*_j - S^H_{q+1} + qc - \lambda^H_j \\
&\Rightarrow a(r+q)c < (2r+q)c - c + 2N + 2\lambda^*_j - S^H_{q+1} - \lambda^H_j \qquad (3.21)
\end{aligned}$$

d'après (3.11) $N < s_{\sigma^H_{N^H_q+1}}$ et étant donné que $J_{\sigma^H_{N^H_q+1}}$ est la première tâche de b^H_{q+1} alors $S^H_{q+1} \geq s_{\sigma^H_{N^H_q+1}}$. De même, les tâches étant ordonnées par ordre croissant des tailles alors $\lambda^H_j \geq s_{\sigma^H_{N^H_q+1}}$. Ainsi, $2N \leq S^H_{q+1} + \lambda^H_j$. De la même manière, λ^*_j est la taille d'un sous-ensemble de tâches du lot b^*_{q+r}, alors $\lambda^*_j \leq c$. Ainsi nous avons.

$$-c + 2N + 2\lambda^*_j - S^H_{q+1} - \lambda^H_j \leq c$$

L'inégalité (3.21) peut alors s'écrire de la façon suivante.

$$\begin{aligned}
a(r+q)c < (2r+q)c + c &\Leftrightarrow a(r+q) < 2r+q+1 \\
&\Leftrightarrow a < \frac{2r+q+1}{r+q} \\
&\Rightarrow a < \frac{2r+2q}{r+q} \\
&\Leftrightarrow a < 2 \qquad (3.22)
\end{aligned}$$

□

Nous avons montré dans ce lemme que deux tâches $J_{\sigma^H_j}$ et $J_{\sigma^*_j}$ produites à la position j et livrées dans les lots b^H_k et b^*_i dans la solution S^H et S^* respectivement sont telles que $k < 2i$. Nous montrons dans le lemme suivant que la somme des dates de livraisons des tâches sur les deux solutions vérifie $\sum_{j=1}^n D^H_{\sigma^H_j} < 2\sum_{j=1}^n D^*_{\sigma^*_j}$.

Lemme 3.2.3 *Les sommes des dates de livraisons optimales et celles obtenues par H vérifient :* $\sum_{j=1}^n D^H_{\sigma^H_j} < 2\sum_{j=1}^n D^*_{\sigma^*_j}$.

Soit $J_{\sigma_j^H}$ une tâche produite en position j et inclue dans le lot b_k^H dans la solution heuristique S^H. La date de livraison de la tâche $J_{\sigma_j^H}$ est égale à $\tau_k + t$. D'après le lemme (3.2.2), dans la solution optimale, la tâche $J_{\sigma_j^*}$ produite en position j est contenue dans le lot b_i^H telle que $k < 2i$. Nous avons :

$$\begin{aligned} D^H_{\sigma_j^H} &= h\mu + t \\ &< 2h\mu + t \\ &< 2D^*_{\sigma_j^*} \end{aligned}$$

Ainsi, la somme des dates de livraisons sur les deux solutions vérifie $\sum_{j=1}^{n} D^H_{\sigma_j^H} < \sum_{j=1}^{n} 2D^*_{\sigma_j^*}$. □

Nous avons montré dans ce théorème que le cas particulier dans lequel les livraisons sont considérées régulières et $s_j = f(p_j)$ admettait une 2-approximation. Nous montrons dans l'observation suivante que le problème n'admet pas d'approximation absolue lorsque les dates de départ des véhicules sont considérées quelconques.

Observation 3.2.4 *Dans le cas où $\mu_i > \mu_{i-1}$, une a-approximation de la solution telle que a soit constant n'est pas possible, à moins que $P = NP$.*

Afin de montrer cela, nous commençons par présenter le problème NP-difficile de 2-partition.

`2-PARTITION`. Etant donné un ensemble S de n entiers $a_1, \ldots, a_n$, existe-t-il $I \subset S$ tel que $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

A partir de l'instance de 2-partition, nous construisons une instance à notre problème tel que $s_j = p_j = f(p_j) = a_j$, $\forall j = 1, \ldots, n$. Soit $c = \sum_{j=1}^{n} s_j/2$ et soient μ_1 et μ_2 les dates des deux premiers départs du véhicule tels que $\mu_1 = \sum_{j=1}^{n} p_j/2$ et $\mu_2 = \sum_{j=1}^{n} p_j$. Soit également μ_3 la date de départ de la troisième tournée telle que $\mu_3 \geq \mu_2$ (à noter qu'il existe une solution réalisable quelque soit l'ordre de production des tâches). Ainsi, le problème admet une a-approximation telle que a soit constant est une solution dans laquelle les tâches sont livrées durant les deux premiers départs et revient à résoudre le problème de 2-partition. Autrement, l'approximation dépend de la date de départ μ_3.

3.3 Formulation PLNE du problème

Dans cette section, nous traitons le cas général $1 \to D, k = 1 | v = 1, c, no\ wait | \sum D_j$ dans lequel les tailles et les durées de productions des tâches sont considérées quelconques. Les dates de départ du véhicule sont également quelconques et connues à l'avance, on note par $T = \{\tau_1, \ldots, \tau_m\}$ l'ensemble des dates de départ supposé être suffisant pour que toutes les tâches soient livrées. Le problème étant NP-difficile, nous présentons deux formulations mathématiques pour le problème, une formulation sous forme de programme linéaire en nombres entiers ainsi qu'une deuxième formulation sous forme de problème de partitionnement d'ensembles adaptée à la génération de colonnes.

3.3.1 Formulation compacte

Nous introduisons dans cette section un modèle de programmation linéaire en nombres entiers pour le problème. On commence par présenter les variables utilisées.

— $y_{j,k} \in \{0,1\}$ égale à 1 si la tâche J_j est dans le lot en position k (b_k), 0 sinon.
— $z_k \geq 0$ indique le temps supplémentaire alloué à la production du lot b_k.
— $L_k \geq 0$ indique la date de fin de production du lot b_k.

Etant donné que la production de b_{k-1} finit au plus tard à la date τ_{k-1}, z_k est le temps de production non consommé par le lot b_{k-1} alloué à b_k et prend comme valeur la différence entre L_{k-1} et τ_{k-1}. i.e., $z_k = \tau_{k-1} - L_{k-1}$.

$$\min \sum_{j=1}^{n}\sum_{k=1}^{m}(\tau_k + t)y_{jk} \tag{3.23}$$

$$\sum_{k=1}^{m} y_{jk} = 1 \qquad \forall j \in \{1,...,n\} \tag{3.24}$$

$$\sum_{j=1}^{n} s_j y_{jk} \leq c \qquad \forall k \in \{1,...,m\} \tag{3.25}$$

$$\sum_{j=1}^{n} p_j y_{jk} \leq \mu_k + z_k \qquad \forall k \in \{1,...,m\} \tag{3.26}$$

$$z_k = z_{k-1} + \mu_{k-1} - \sum_{j=1}^{n} p_j y_{jk-1} \qquad \forall k \in \{2,...,m\} \tag{3.27}$$

$$z_1 = 0 \tag{3.28}$$

$$y_{j,k} \in \{0,1\} \qquad \forall j \in \{1,...,n\}, \forall k \in \{1,...,m\} \tag{3.29}$$

$$z_k \geq 0 \qquad \forall k \in \{2,...,m\} \tag{3.30}$$

Le premier ensemble de contraintes (3.24) nous assure qu'une tâche est contenue dans exactement un lot. Les contraintes (3.25) garantissent que la capacité du véhicule n'est pas dépassée. Le temps disponible pour produire le premier lot étant égal à $\tau_1 = \mu_1$, le temps supplémentaire alloué à la production du premier lot est fixé à zéro (3.28). Les contraintes (3.26) garantissent que la production d'un lot finit au plus tard à sa date de départ pour la livraison. Le temps supplémentaire z_k alloué à la production d'un lot b_k étant égal au temps non consommé par b_{k-1}, les contraintes (3.27) retournent la différence entre la date de fin de production du lot b_{k-1} et sa date de départ. Une tâche J_j contenue dans un lot b_k étant livrée à la date $\tau_k + t$, l'objectif à minimiser (3.23) retourne la somme des dates de livraison des tâches.
Ce modèle contient nm variables binaires, n variables continues et $O(nm)$ contraintes.

3.3.2 Formulation étendue

Dans cette section, nous présentons une formulation sous forme de partition d'ensembles pour le problème maître. Une colonne représente un lot et l'ensemble des lots réalisables

est noté β. Soit $\beta^k \in \beta$ l'ensemble des lots réalisables à une position k, i.e., les lots partant à la date τ_k et $b_i^k \in \beta^k$ le i^e lot de β^k (i.e., $\beta^k = \{b_1^k, \ldots, b_i^k, \ldots\}$ pour tout $k \in \{1, \ldots, m\}$). Pour tout lot $b_i^k \in \beta$, la variable (colonne) du problème lui correspondant est définie comme suit.

$$x_i^k = \begin{cases} 1 & \text{si le lot } b_i^k \text{ est sélectionné} \\ 0 & \text{sinon} \end{cases}$$

Une fois un lot sélectionné, les tâches qu'il contient ainsi que sa durée d'exécution sur la machine sont connus. Les notations suivantes sont utilisées dans le modèle.

— $a_j^{ki} = 1$ si la tâche J_j est contenue dans le lot b_i^k, 0 sinon
— n_i^k indique le nombre de tâches contenues dans le lot b_i^k correspondant à la colonne x_{ik}
— P_i^k la durée de production du lot b_i^k

$$\min \sum_{k=1}^{n} \sum_{i \in \beta^k} n_i^k \tau_k x_i^k \tag{3.31}$$

$$\sum_{k=1}^{l} \sum_{i \in \beta^k} P_i^k x_i^k \leq \tau_l \qquad \forall l \in \{1, ..., m\} \tag{3.32}$$

$$\sum_{k=1}^{m} \sum_{i \in \beta^k} a_j^{ki} x_i^k = 1 \qquad \forall j \in \{1, ..., n\} \tag{3.33}$$

$$x_i^k \in \{0, 1\} \qquad \forall k \in \{1, ..., m\}, \forall i \in \beta^k \tag{3.34}$$

Le premier ensemble de contraintes (3.32) garantit que la production d'un lot est finie avant sa date départ pour la livraison. Etant donné que l'ordre de production des lots et l'ordre de livraison sont les mêmes, si $k_1 < k_2$ alors le lot sélectionné de l'ensemble β^{k_1} doit être produit avant celui sélectionné de β^{k_2} et au k^e départ du véhicule, la production des lots $\beta^1, \ldots, \beta^k$ doit être finie. Les contraintes (3.33) nous assurent que toutes les tâches sont affectées à exactement un lot. L'objectif à minimiser est donné par (3.31).

Suivant le schéma standard de la génération de colonnes, le problème maître est restreint à un sous-ensemble de variables (colonnes) $\tilde{\beta} \subseteq \beta$. Etant donné une solution optimale de la relaxation continue du problème restreint au variables $\tilde{\beta}$, la résolution d'un sous-problème est nécessaire pour trouver des éléments de $\beta \setminus \tilde{\beta}$ pouvant améliorer la solution et les introduire au problème maître. La solution de la relaxation continue du problème restreint aux $\tilde{\beta}$ variables est dite optimale si aucun élément de $\beta \setminus \tilde{\beta}$ n'améliore la solution.

Sous-problème (pricing problem) Le sous-problème cherche un élément de $\beta \setminus \tilde{\beta}$ tel que le coût réduit de la nouvelle colonne soit négatif. On note par α_l et γ_j les variables duales associées aux contraintes (3.32) et (3.33) respectivement. Le coût réduit de la variable x_i^k

noté $\bar{c}_i^k$ est comme suit.

$$\begin{aligned}\bar{c}_i^k &= n_i^k \tau_k - \sum_{l=k}^{m} P_i^k \alpha_l - \sum_{j=1}^{n} a_j^{b_i^k} \gamma_j, \quad (P_i^k = \sum_{j=1}^{n} p_j a_j^{b_i^k} \; et \; n_i^k = \sum_{j=1}^{n} a_j^{b_i^k}) \\ &= \sum_{j=1}^{n} a_j^{b_i^k} \tau_k - \sum_{l=k}^{m} \sum_{j=1}^{n} p_j a_j^{b_i^k} \alpha_l - \sum_{j=1}^{n} a_j^{b_i^k} \gamma_j, \\ &= \sum_{j=1}^{n} \underbrace{(\tau_k - p_j \sum_{l=k}^{m} \alpha_l - \gamma_j)}_{w_j} a_j^{b_i^k}\end{aligned}$$

Soit $w_j = \tau_k - p_j \sum_{l=k}^{m} \alpha_l - \gamma_j$, le sous-problème peut être formulé de la manière suivante. Un ensemble de variables $z_j \in \{0,1\}, j \in \{1,\ldots,n\}$ où z_j prend la valeur 1 si la tâche J_j est inclue dans le nouveau lot, 0 sinon.

$$\min \sum_{j=1}^{n} w_j z_j \tag{3.35}$$

$$\sum_{j=1}^{n} s_j z_j \leq c \tag{3.36}$$

Etant donné que le lot retourné doit être réalisable, la contrainte (3.36) nous garantit que la capacité du véhicule n'est pas dépassée.

Clairement, dans une solution optimale, les variables $z_j, j \in \{1,\ldots,n\}$ telles que $w_j < 0$ prennent la valeur zéro. Par conséquent, nous considérons uniquement les variables telles que $w_j \geq 0$. Soit J' le sous-ensemble de tâches restantes et $w'_j = -w_j$. Le problème qui en resulte est un problème de sac-à-dos.

$$\max \sum_{j=1}^{n'} w'_j z_j \tag{3.37}$$

$$\sum_{j=1}^{n'} s_j z_j \leq c \qquad \forall j \in \{1,...,n'\} \tag{3.38}$$

3.4 Conclusion

Nous avons présenté dans ce chapitre un cas déterministe du problème intégré d'ordonnancement et de distribution où les tâches sont produites sur une seule machine et les produits finis sont livrés vers un unique client par des véhicules de capacités identiques dont les dates de départ sont fixes et connues à l'avance par le fabricant. A chaque tâche sont associés une durée de production et un espace occupé dans le véhicule et l'objectif est de minimiser la somme des dates de livraisons des tâches. Le problème est montré NP-difficile et un algorithme d'approximation est proposé pour un cas particulier. Une formulation compacte du problème ainsi qu'une formulation adaptée à la génération de colonnes sont également proposées.

En perspective, nous effectuerons des expérimentations afin de valider les modèles proposés et nous proposerons un branch & bound afin d'obtenir des solutions optimales du problème ainsi que des algorithmes d'approximation pour des cas particuliers moins restreints.

Chapitre 4

Problème général de production et de distribution intégrées : complexité et génération de colonnes

Dans ce chapitre, nous étudions un cas général du problème intégré de production et de distribution déterministe. Les tâches sont produites par une machine et livrées vers plusieurs clients situés à des localisations géographique différentes. Un unique véhicule de capacité limitée est utilisé pour effectuer la livraison. Le problème consiste alors à déterminer la séquence de production des tâches, la constitution des lots et la séquence de livraison des tâches de chaque lot. La partie distribution est traitée de manière explicite dans ce chapitre et ainsi le problème traité constitue une variante de voyageur de commerce à tournées multiples. Deux fonctions objectif très utilisées en ordonnancement sont considérées dans ce chapitre, la première consiste à minimiser le makespan, i.e., la date de retour du véhicule au dépôt après que toutes les tâches soient livrées. La seconde consiste à minimiser la somme des dates de livraison des tâches. Des résultats de complexité sur des cas particuliers sont proposés et une approche par génération de colonnes est proposée pour résoudre les relaxations continues d'une formulation étendue du problème pour chacune des fonctions objectifs. Enfin, des résultats expérimentaux sont présentés afin d'attester les bonnes performances des méthodes proposées. Ce chapitre est issu d'une collaboration avec S.U. Ngueveu et a été présenté dans la conférence internationale ISCO 2016 et sélectionné pour une publication dans *Lecture Notes in Computer Science* [Cheref *et al.*, 2016d].

4.1 Description du problème et notations

Dans ce chapitre, les tâches sont produites sur une machine et la préemption sur la production d'une tâche n'est pas autorisée. A chaque tâche $J_j \in \{1, \ldots, n\}$ sont associées une durées d'exécution p_j, une taille occupée dans le véhicule s_j et une destination j. La partie livraison est traitée par un véhicule de capacité c. Les coûts de livraison n'étant pas pris en compte dans cette thèse, des indicateurs de temps sont utilisés afin de mesurer la qualité des solutions. Ainsi, dans la première fonction objectif on cherche à minimiser le

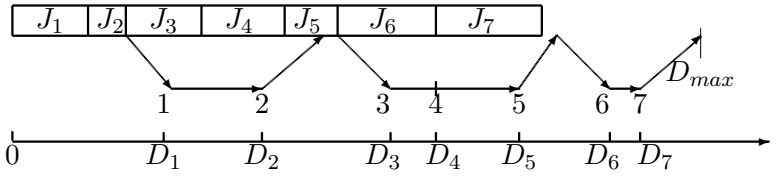


FIGURE 4.1 – Illustration des problèmes $1 \rightarrow D, k \geq 1 | v = 1, c | D_{max}$ ou $\sum D_j$

temps total nécessaire afin de compléter la production et la livraison, ce critère étant noté D_{max}. Le second critère consiste à minimiser la date de livraison moyenne (ou somme des dates de livraisons) des tâches. A noter que D_{max} est la date de retour du véhicule après avoir livré la dernière tâche (voir la figure 4.1).

Etant donné que la partie tournées de véhicules fait partie de la décision, ce problème peut être considéré comme étant une variante du problème de tournées de véhicules. En effet, lorsque l'objectif est de minimiser la date de fin totale de la production et de la livraison, le problème s'apparente au problème de tournées de véhicules dans lequel un véhicule peut effectuer plusieurs tournées noté *"multi-trip vehicle routing problem"* (voir [Azi *et al.*, 2007], [Azi *et al.*, 2010], [Macedo *et al.*, 2011], [Hernandez *et al.*, 2016] pour plus de détails sur ce problème). Ainsi dans le problème considéré dans la littérature, les tâches sont supposées être disponibles à l'instant 0 et le problème consiste finalement à déterminer les tournées de sorte à minimiser la date de fin de livraison de toutes les tâches en respectant des contraintes particulières aux problèmes de tournées de véhicules telles que les fenêtres de temps de livraison, une longueur maximum pour les tournées, etc. Dans [Azi *et al.*, 2010], [Macedo *et al.*, 2011], [Hernandez *et al.*, 2016], les auteurs proposent une procédure de génération de colonnes pour le problème. Cependant, Cattaruzza et al dans [Cattaruzza *et al.*, 2016] considèrent que les tâches sont disponibles à des dates données mais sans pour autant que le problème d'ordonnancement ne fasse partie de la décision. Les auteurs proposent une heuristique pour la résolution du problème. D'autre part, le problème d'ordonnancement et de distribution dans lequel l'objectif considéré est de type cumulatif, i.e., la somme des dates de livraisons, constitue une variante du problème de tournée de véhicules à dates de livraisons cumulatives (*"cumulative capacitated vehicle routing problem"*). Une méthode de génération de colonne est également proposée pour ce problème par Lysgaard et Wøhlk [Lysgaard et Wøhlk, 2014].

Enfin, le problème considéré est noté $1 \rightarrow D, k \geq 1 | v = 1, c | D_{max}$ lorsque l'objectif D_{max} est considéré et $1 \rightarrow D, k \geq 1 | v = 1, c | \sum D_j$ pour la somme des dates de livraisons. Une illustration de ces deux problèmes avec $n = 7$ est donnée dans la figure (4.1).

Dans ce qui suit, nous commençons par une étude de complexité sur des cas particuliers du problème et proposons une formulation étendue pour le cas général adaptée au schéma de la génération de colonnes. Les objectif D_{max} et $\sum D_j$ sont considérés dans chacun des cas.

4.2 Cas particuliers

Dans cette section, nous considérons deux cas particuliers du problème pour chacune des deux fonctions objectif. Nous commençons par donner quelques résultats de complexité sur

le cas particulier avec un seul client (site de livraison) ainsi que des observations concernant le cas où les lots sont fixés, i.e. la composition des lots est prédéterminée.

4.2.1 Cas avec un seul client

4.2.1.1 Problème $1 \to D, k = 1|v = 1, c|D_{max}$

Dans [Chang et Lee, 2004], les auteurs montrent que le problème $1 \to D, k = 1|v = 1, c|D_{max}$ est équivalent au problème du Bin Packing lorsque les durées d'exécution des tâches sont nulles. En effet, une solution optimale de ce cas particulier du problème sous l'objectif D_{max} consiste à minimiser le nombre de tournées et ceci revient à minimiser le nombre de boites dans un problème de Bin Packing. Ainsi, le problème $1 \to D, k = 1|v = 1, c|D_{max}$ est NP-difficile au sens fort. A noter que ce raisonnement n'est pas valide sur l'objectif $\sum_j D_j$. En effet, une solution avec un minimum de tournées n'est pas forcément optimale lorsque l'objectif est la somme des dates de livraisons.

4.2.1.2 Problème $1 \to D, k = 1|v = 1, c|\sum_j D_j$

La preuve utilisée pour l'objectif D_{max} n'étant pas valide sur ce deuxième objectif, nous montrons que le problème $1 \to D, k = 1|v = 1, c|\sum_j D_j$ est également NP-difficile au sens fort par une réduction polynomiale au problème de `3-PARTITION`. La preuve du théorème qui suit est similaire à celle utilisée dans le chapitre 3 pour la preuve de complexité du problème $1 \to D, k = 1|v = 1, c = z, no\ wait|\sum_j D_j$ (voir 3.1.3). A noter que la démonstration proposée dans la section 3.1.3 reste valable pour le problème $1 \to D, k = 1|v = 1, c|\sum_j D_j$. Cependant, nous proposons dans ce qui suit une preuve plus générale dans laquelle les durée de production sont nulles.

Théorème 4.2.1 *Le problème $1 \to D, k = 1|v = 1, c = z|\sum_j D_j$ est NP-difficile au sens fort.*

Démonstration. Nous commençons par présenter le problème fortement NP-difficile de `3-PARTITION`. Etant donné un ensemble de $3h$ entiers $a_1, \ldots, a_{3h}$ telles que $\sum a_i = hb$ et $b/4 < a_i < b/2$ pour tout i, le problème consiste à partitionner cet ensemble d'entiers en h ensembles disjoints tels que la somme de chaque sous ensemble soit égale à b.

Ainsi, étant donnée une instance de `3-PARTITION`, on construit une instance de notre problème de la manière suivante :

— $n = 3h$ tâches et la capacité $c = b$,
— Les temps de transport $t_{M1} = t_{1M} = t$ et $t > 0$, quelconque
— Pour tout $J_j \in \{J_1, \ldots, J_{3h}\}$ $p_j = 0, s_j = a_j$,
— La somme des dates de livraisons $y = \frac{3}{2}h(h+1)bt$.

De là, le problème consiste à déterminer s'il existe une solution telle que $\sum_j D_j \leq y$.

De `3-PARTITION` à $1 \to D, k = 1|v = 1, c|\sum_j D_j$. *S'il existe une solution à l'instance de `3-PARTITION` alors il existe pour notre problème une solution réalisable S telle que $\sum_j D_j \leq y$.*

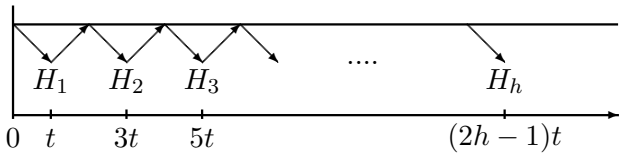


FIGURE 4.2 – Solution de `3-PARTITION`

Soient $H_1, H_2, \ldots, H_h$ une solution de l'instance `3-PARTITION`. On construit une solution S pour le problème $1 \rightarrow D, k = 1 | v = 1, c | \sum_j D_j$ telle que $\sum_j D_j = \frac{3}{2}h(h+1)bt = y$. On construit une solution à notre problème en affectant le sous-ensemble de tâches H_i au lot b_i pour tout $i \in \{1, \ldots, h\}$. Le véhicule commence la première tournée à la date 0 et ainsi les trois tâches de H_1 sont livrées à la date t et le véhicule retourne au dépôt à la date $2t$. Etant donné que les durées de production sont nulles, le véhicule repart immédiatement avec le second lot constitué des tâches de H_2 et le livre à la date $3t$. En suivant le même raisonnement (voir Fig. 4.2), un lot b_i est alors livré à la date $(2i-1)t$ et la somme des dates d'arrivées est égale à $\sum_j D_j = 3\sum_{i=1}^{h}(2i-1)t = y$.

De $1 \rightarrow D, k = 1 | v = 1, c | \sum_j D_j$ à `3-PARTITION`. Inversement, on montre que *s'il existe une solution S au problème $1 \rightarrow D, k = 1 | v = 1, c | \sum_j D_j$ telle que $\sum_j D_j \leq y$ alors l'instance de `3-PARTITION` admet une solution.* La preuve de la réciproque étant équivalente à celle de la preuve de complexité du problème $1 \rightarrow D, k = 1 | v = 1, c = z, no\ wait | \sum_j D_j$, nous renvoyons à la section 3.1.3. □

Remarque Le problème $1 \rightarrow D, k = 1 | v = 1, c | \sum_j D_j$ peut être résolu en temps polynomial dans le cas où toutes les tâches ont la même taille [Li *et al.*, 2005]. Les auteurs proposent un algorithme de programmation dynamique avec une complexité en $O(n^2)$ pour résoudre le problème.

4.2.2 Cas avec des lots prédéterminés

Dans ce deuxième cas particulier, nous considérons que les tâches sont préalablement affectées aux lots et par conséquent les tournées correspondant aux lots sont également connues. Pour chacune des deux fonctions objectif, ce cas particulier du problème est traité.

4.2.2.1 Problème $1 \rightarrow D, k \geq 1 | v = 1, c, fixed - batchs | D_{max}$

Proposition 4.2.2 *Le problème $1 \rightarrow D, k \geq 1 | v = 1, c = z, fixed - batchs | D_{max}$ peut être résolu en temps polynomial.*

La composition des lots ainsi que les tournées étant prédéterminées, un lot peut alors être considéré comme étant une tâche et ainsi ce problème devient équivalent au célèbre problème de minimisation du temps de production total sur un flow shop à deux machines.

`Flow Shop`. Etant donné un ensemble de n tâches et un ensemble de m machines, trouver une séquence de production qui optimise un critère d'évaluation en respectant les

contraintes suivantes.

— Les tâches doivent être traitées sur toutes les machines, de la machine 1 à m,

— La machine ne traite qu'une seule tâche à la fois.

Le flow shop à deux machines avec minimisation du makespan (noté $F2||C_{max}$) peut être résolu en temps polynomial par l'algorithme de Johnson [Johnson, 1954].

Ainsi, la résolution du problème $1 \rightarrow D, k \geq 1|v = 1, c = z, fixed - batchs|D_{max}$ est équivalente à la résolution d'un flow shop à deux machines avec comme objectif la minimisation du makespan. Ceci s'obtient en considérant un lot du problème $1 \rightarrow D, k \geq 1|v = 1, c = z, fixed - batchs|D_{max}$ comme étant une tâche du flow shop à deux machines, la durée de production et la durée de la tournée du lot comme étant respectivement les durées de production de la tâche correspondante sur la première et la seconde machine.

4.2.2.2 Problème $1 \rightarrow D, k = 1|v = 1, c, fixed - batchs| \sum_j D_j$

Nous montrons dans ce qui suit que le cas particulier avec lot fixé est NP-difficile par une réduction polynomiale au problème de flow shop à deux machines. A noter que celui-ci est NP-difficile lorsque le critère d'optimisation est la minimisation de la somme des dates de fin de production noté $F2|| \sum_j C_j$ [Garey *et al.*, 1976].

Proposition 4.2.3 *Le problème* $1 \rightarrow D, k \geq 1|v = 1, c = z, fixed - batchs| \sum_j D_j$ *est NP-difficile.*

Puisque les lots sont prédéfinis, nous considérons le cas particulier où chaque lot contient exactement une seule tâche. Soit C'_j la date à laquelle le véhicule retourne au dépôt après avoir livré la tâche J_j, i.e., $C'_j = D_j + t_{jM}$. Donc la date de livraison de la tâche J_j est $D_j = C'_j - t_{jM}$ et la somme des dates de livraisons $\sum_{j=1}^{n} D_j = \sum_{j=1}^{n} C'_j - \sum_{j=1}^{n} t_{jM}$. Etant donné que $\sum_{j=1}^{n} t_{jM}$ est constant, minimiser $\sum_{j=1}^{n} D_j$ revient à minimiser $\sum_{j=1}^{n} C'_j$. Ainsi, de même que pour l'objectif D_{max}, ce cas particulier du problème $1 \rightarrow D, k = 1|v = 1, c, fixed - batchs| \sum_j D_j$ est équivalent à un flow shop à deux machines. En effet, en considérant p_j comme la durée de la tâche sur la première machine et la durée de la tournée qui est égale à $t_{Mj} + t_{jM}$ comme étant celle sur la deuxième machine, résoudre ce cas particulier revient à résoudre le problème NP-difficile $F2|| \sum_j C'_j$ où C'_j correspond à la date de fin de production de J_j sur la deuxième machine.

4.3 Cas général

Nous abordons dans cette section un cas général du problème intégré d'ordonnancement et de tournées de véhicules dans lequel le problème d'ordonnancement à une machine ainsi que le problème de tournées de véhicules font partie de la décision. Cependant, avant d'entamer les méthodes de résolution, nous établissons quelques propriétés fondamentales de dominance entre solutions. Du fait que les dates de début au plus tôt ne sont pas considérées dans cette première partie de la thèse, à partir de n'importe quelle solution, la séquence de production peut facilement être changée et réordonnée suivant la séquence de livraison des tâches et cela sans affecter le critère d'optimisation (D_{max} et $\sum_j D_j$). Les propriétés sont alors énoncées comme suit.

Propriétés 4.3.1 *Il existe une solution optimale au problème* $1 \rightarrow D, k \geq 1 | v = 1, c = z | D_{max}$ *ou* $\sum_j D_j$ *qui satisfait les conditions suivantes :*

- *Les tâches sont ordonnancées sur la machine sans temps d'arrêt,*
- *La séquence de production et la séquence de livraison des tâches sont les mêmes.*

Les propriétés énoncées impliquent que lorsque la tournée livrant un lot est connu, la séquence de production du lot sur la machine peut alors être déduite. Pour la fonction objectif D_{max} et $\sum_j D_j$, nous pouvons voir que la solution optimale du problème $1 \rightarrow D, k \geq 1 | v = 1, c = z | D_{max}$ *ou* $\sum_j D_j$ peut être obtenue en combinant les lots et ainsi le problème est adapté à une modélisation sous forme de partition d'ensembles. Par conséquent une formulation étendue adaptée à la génération de colonnes peut être proposée pour chacune des deux fonctions objectifs. A noter que le problème considéré sans la phase ordonnancement est similaire à un problème de tournée de véhicules à plusieurs tournées (*multi trip vehicle routing problem*) pour lequel la génération de colonnes et le branch & price sont les méthodes de choix pour une résolution exacte [Azi *et al.*, 2010],[Lysgaard et Wøhlk, 2014]. Cependant, nous disposons dans notre cas d'un seul véhicule et d'une phase d'ordonnancement qui contraint la constitution des lots. Il est donc pertinent de se demander si une approche de génération de colonnes peut toujours être appliquée avec succès sur le problème considéré.

4.3.1 Problème $1 \rightarrow D, k \geq 1 | v = 1, c | D_{max}$

Le concept de génération de colonnes étant présenté dans le chapitre précédent (voir chapitre 3). Nous pouvons dés à présent introduire une formulation sous forme de partition d'ensemble pour le problème maître. Une colonne représente un lot ainsi que sa position dans la séquence de livraison. Les notations utilisées sont comme suit.

- $\beta = \{b_1, \ldots, b_{|\beta|}\}$ l'ensemble des lots réalisables pour lesquels les positions sont fixées,
- $P_{b,1} = \sum_{j \in b} p_j$ la durée de production du lot b sur la machine,
- $P_{b,2}$ la durée de la tournée qui livre le lot b,
- $a_{i,b} = 1$ indique si la tâche J_i est dans la lot b et par conséquent prend la valeur 1, 0 sinon.

En effet, un lot étant un ensemble de tâches dont l'ordre de production et de livraison est le même (d'après les propriétés 4.3.1), une fois celui-ci connu, sa durée de production sur la machine et la tournée qui le livre sont également connues. Un unique ensemble des variables défini pour minimiser D_{max}.

- $x_{b,k}$ vaut 1 si le lot b en position k est sélectionné, 0 sinon.

$$\min \ D_{max} \tag{4.1}$$

$$D_{max} \geq \sum_{k=1}^{l} \sum_{b \in \beta} P_{b,1} x_{b,k} + \sum_{k=l}^{n} \sum_{b \in \beta} P_{b,2} x_{b,k}, \qquad \forall l \in \{1, ..., n\} \tag{4.2}$$

$$\sum_{b \in \beta} x_{b,k} \leq 1, \qquad \forall k \in \{1, ..., n\} \tag{4.3}$$

$$\sum_{b \in \beta} (a_{i,b} \sum_{k=1}^{n} x_{b,k}) = 1, \qquad \forall i \in \{1, ..., n\} \tag{4.4}$$

$$\sum_{b \in \beta} x_{b,k} \geq \sum_{b \in \beta} x_{b,k+1}, \qquad \forall k \in \{1, ..., n-1\} \tag{4.5}$$

$$\sum_{b \in \beta} x_{b,k} = 1, \qquad \forall k \in \{1, ..., \delta\} \tag{4.6}$$

$$x_{b,k} \in \{0, 1\} \qquad \forall k \in \{1, ..., n\}, \forall b \in \beta \tag{4.7}$$

Le premier ensemble de contraintes (4.2) exprime le flow shop à deux machines présenté précédemment sur le cas avec lots fixés (voir 4.2.2.1). Ces contraintes garantissent que la production d'un lot sur la machine commence après la production du lot qui le précède mais également que le véhicule commence la livraison d'un lot une fois que la production de celui-ci est terminée et que le véhicule est retourné au dépôt. Les contraintes (4.3) et (4.4) garantissent la réalisabilité de la solution en imposant au plus un lot à chaque position (4.3) et en s'assurant que chaque tâche est affectée à un seul lot (4.4). Les contraintes (4.5) sont utilisées afin d'éliminer la symétrie en forçant les colonnes (lots) sélectionnées à apparaître consécutivement aux premières positions. Les contraintes (4.6) sélectionnent un nombre minimum de positions par l'intermédiaire d'une borne inférieure du nombre minimum de lots nécessaire pour contenir toutes les tâches.

Cette borne inférieure est notée δ et afin de l'obtenir, nous utilisons la règle `First Fit Decreasing` qui est une 3/2 approximation pour le problème du bin-packing (voir le chapitre 3 pour plus de détail sur le `FFD`). Ainsi en considérant uniquement la taille des tâches, soit τ le nombre de boites obtenues par l'algorithme `FFD`. Le nombre minimum de lots nécessaires δ est alors égal à $2/3\tau$. A noter que les contraintes (4.5) et (4.6) constituent des inégalités valides et par conséquent ne sont pas indispensables au programme linéaire en nombre entier.

Le problème maître étant établi, le schéma standard de la génération de colonnes implique que celui-ci soit au préalable restreint à un sous-ensemble de colonnes (variables) notées $\tilde{\beta} \subseteq \beta$. En effet, le nombre de combinaisons de lots total étant très grand, le problème maître est restreint à un sous ensemble de variables et un sous-problème est utilisé afin de trouver des éléments de $\beta \setminus \tilde{\beta}$ pouvant améliorer la solution la relaxation continue du problème maître restreint (représenté par les contraintes (4.2-4.6)) et les introduire à ce dernier.

Sous-problème (pricing problem) Soient $\alpha_l, \beta_k, \gamma_i, \sigma_k$ et ξ_k les variables duales associées respectivement aux contraintes (4.2), (4.3),(4.4), (4.5), (4.6) de la relaxation continue

du problème maître restreint. A noter que la variable duale ξ_k existe uniquement pour $k \leq \delta$ et la variable σ_{k-1} pour $k \geq 2$ et σ_k for $k \leq n-1$. On note par $\bar{c}_{b,k}$ le coût réduit de la variable $x_{b,k}$ et ainsi le sous-problème cherche un élément de $\beta \setminus \tilde{\beta}$ tel que $\bar{c}_{b,k} < 0$.

$$\begin{aligned}
\bar{c}_{b,k} &= \sum_{l=k}^{n} P_{b,1}\alpha_l + \sum_{l=1}^{k} P_{b,2}\alpha_l - \beta_k - \sum_{i=1}^{n} a_{i,b}\gamma_i + \sigma_{k-1} - \sigma_k - \xi_k, \quad (P_{b,1} = \sum_{i=1}^{n} p_i a_{i,b}) \\
&= \sum_{i=1}^{n}\sum_{l=k}^{n} p_i a_{i,b}\alpha_l - \sum_{i=1}^{n} a_{i,b}\gamma_i + \sum_{l=1}^{k} P_{b,2}\alpha_l - \beta_k + \sigma_{k-1} - \sigma_k - \xi_k \\
&= \sum_{i=1}^{n} \frac{p_i \sum_{l=k}^{n} \alpha_l - \gamma_i}{\sum_{l=1}^{k} \alpha_l} a_{i,b} + P_{b,2} - \frac{\beta_k - \sigma_{k-1} + \sigma_k + \xi_k}{\sum_{l=1}^{k} \alpha_l}
\end{aligned}$$

donc

$$\bar{c}_{b,k} < 0 \quad \Leftrightarrow \sum_{i=1}^{n} \underbrace{\frac{p_i \sum_{l=k}^{n} \alpha_l - \gamma_i}{\sum_{l=1}^{k} \alpha_l}}_{\ell_i} a_{i,b} + P_{b,2} < \underbrace{\frac{\beta_k - \sigma_{k-1} + \sigma_k + \xi_k}{\sum_{l=1}^{k} \alpha_l}}_{r_k} \tag{4.8}$$

Nous définissons un graphe orienté auxiliaire $G = (V, A)$ dans lequel l'ensemble des noeuds est noté $V = N \cup \{v_s, v_d\}$ où les noeuds $N = \{v_1, \ldots, v_n\}$ représentent les n sites de livraisons du problème $1 \to D, k \geq 1 | v = 1, c | D_{max}$ et $\{v_s, v_d\}$ représentent le dépôt dupliqué. L'ensemble des arcs du graphe G est noté $A = \{(v_i, v_j) | v_i \in N \cup \{v_s\}, v_j \in N \cup \{v_d\}, v_i \neq v_j\}$. Sans perte de généralité, nous considérons qu'il existe un chemin direct entre tous les sites de livraison et on note par $A = \{(v_i, v_j) | v_i \in N \cup \{v_s\}, v_j \in N \cup \{v_d\}, v_i \neq v_j\}$ l'ensemble des arcs du graphe G. Enfin, d'après l'expression (4.8) du coût réduit, les notations suivantes sont utilisées afin de simplifier le sous-problème.

$$\begin{aligned}
\ell_i &= \frac{p_i \sum_{l=k}^{n} \alpha_l - \gamma_i}{\sum_{l=1}^{k} \alpha_l}, && \forall i \in \{1, ..., n\} \\
r_k &= \frac{\beta_k - \sigma_{k-1} + \sigma_k + \xi_k}{\sum_{l=1}^{k} \alpha_l}, && \forall k \in \{1, ..., n\}
\end{aligned}$$

Dès lors le coût réduit peut être écrit de la manière suivante.

$$\bar{c}_{b,k} < 0 \quad \Leftrightarrow \sum_{i=1}^{n} \ell_i a_{i,b} + P_{b,2} < r_k \tag{4.9}$$

où les $a_{i,b}$ indiquent l'ensemble des tâches sélectionnées pour le lot b et $P_{b,2}$ la durée de la tournée qui livre b. Soit d_{ij} la distance de l'arc (v_i, v_j). Ainsi, la distance entre deux noeuds (v_i, v_j) est égale à $\ell_i + t_{ij}$ pour tout $v_i \in V - \{v_s\}$. En effet, la constante ℓ_i doit être comptabilisée lorsque le site i est visité et peut alors être prise en compte directement en la rajoutant à chaque arc sortant de v_i. La distance du dépôt jusqu'aux sites de livraison est notée $d_{sj} = t_{Mj}$ pour tout (v_s, v_j) et la distance de l'arc (v_s, v_d) est notée $d_{sd} = r_k$.

En conséquence, le sous-problème consiste à résoudre un problème de plus court chemin

élémentaire avec contraintes de ressources (*Elementary Shortest Path Problem with Resource Constraints*, ESPPRC) sur le graphe G où les contraintes de ressources concernent la capacité c du véhicule.

ESPPRC. Ce problème consiste à trouver le plus court chemin élémentaire entre un point de départ et un point d'arrivée en prenant en compte des contraintes additionnelles représentées sous forme de ressources. En présence de ces contraintes, le problème du plus court chemin élémentaire est montré NP-difficile au sens fort par Dror [Dror, 1994]. Cependant, celui-ci est utilisé avec succès comme sous-problème (*pricing problem*) dans de nombreuses applications du problème de tournée de véhicules [Feillet *et al.*, 2004], [Azi *et al.*, 2010], etc.

Ainsi, la longueur du plus court chemin retourné correspond à la valeur minimum sur toutes les combinaisons de lots de l'équation suivante.

$$\min_{b \in \beta \setminus \tilde{\beta}} \{ \sum_{i=1}^{n} \ell_i a_{i,b} + P_{b,2} \}$$

Par conséquent, la colonne x_{bk} correspondant au plus court chemin retourné peut être introduite au problème maître restreint si la longueur du plus court chemin est inférieure à r_k. A partir du chemin retourné et du fait que la séquence de production est identique à la séquence de livraison, la durée de production du lot sur la machine $P_{b,1}$ et la durée de la tournée qui le livre $P_{b,2}$ sont connues. Afin de résoudre le problème ESPPRC, nous avons utilisé la méthode exacte proposée par Lozano et al. Pour plus de détails sur la méthode, voir [Lozano *et al.*, 2015].

A partir de la première position ($k = 1$), le sous-problème cherche une nouvelle colonne à rajouter au problème maître restreint en testant toutes les positions et s'arrête lorsqu'un coût réduit négatif est trouvé. La nouvelle colonne est ainsi ajoutée au problème maître restreint et le processus est répété jusqu'à ce qu'il n'existe plus de colonne de coût réduit négatif à introduire. Nous proposons dans ce qui suit une heuristique afin d'initialiser le processus de génération de colonnes.

Solution initiale heuristique

Afin d'initialiser le processus de génération de colonnes en prenant en compte les particularités du problème considéré, nous proposons l'heuristique gloutonne suivante.

(1) La première étape consiste à affecter les tâches aux lots suivant leurs tailles. La règle `FFD` du bin-packing est utilisée à cet effet
(2) Les lots étant constitués, la séquence de livraison des tâches d'un lot est déterminée grâce à la règle du plus proche voisin.
(3) Les lots ainsi que leurs durées de production et de livraison étant connus, déterminer la séquence optimale de production et de livraison revient à résoudre le flow-shop à deux machines présenté dans la section (4.2.2.1) en utilisant la règle de Johnson.

A noter que l'heuristique proposée minimise le nombre de positions initiales. Cela permet

d'accélérer la résolution du sous-problème vu que le nombre de positions est relativement petit.

4.3.2 Problème $1 \rightarrow D, k \geq 1 | v = 1, c | \sum_j D_j$

Nous proposons dans cette section une nouvelle formulation étendue dans laquelle la fonction objectif cumulative $\sum_j D_j$ est traitée. Afin d'obtenir la date de livraison pour chaque tâche, le problème maître doit tenir compte de la date de départ de chaque lot. Les notations utilisées sur ce modèle sont comme suit.

- $\beta' = \{b_1, \ldots, b_{|\beta'|}\}$ l'ensemble des lots réalisables pour lesquels les positions et les dates de départ du véhicule afin de les livrer sont fixées,
- $P_{b,1} = \sum_{j \in b} p_j$ la durée de production du lot b sur la machine,
- $P_{b,2}$ la durée de la tournée qui livre le lot b,
- $a_{i,b} = 1$ si la tâche J_i est dans la lot b et 0 sinon.
- $R_{i,b}$ le temps de transport de la tâche J_i du lot b. Cette valeur constitue le temps écoulé entre la date de départ du véhicule pour livrer b et la date d'arrivée au site de la tâche J_i.

A noter qu'une fois un lot b connu, sa durée de production sur la machine $P_{b,1}$, la durée de la tournée qui le livre $P_{b,2}$, les tâches qui le constituent $a_{i,b} = 1$ tels que $J_j \in b$ et la durée de transport $R_{i,b}$ jusqu'aux différents sites $J_i \in b$ sont connues. Les variables binaires et continues suivantes sont utilisées afin de minimiser $\sum_j D_j$.

- $y_{bkt} = 1$ si le lot b livré en position k et dont la tournée commence à la date t est sélectionné, 0 sinon.
- $S_k \geq 0$ la date de départ du lot en position k.

Soit T une borne supérieure de la date de départ du véhicule pour effectuer la dernière tournée. Du moment que l'inégalité triangulaire est respectée, cette borne est donnée par l'équation suivante.

$$T = \sum_{i=1}^{n} (p_i + 2t_{0,i})$$

A noter que les variables indiquant les dates de livraison de tâches ne sont pas utilisées. Celles-ci sont calculées implicitement dans la fonction objectif du modèle suivant.

$$\text{Minimize} \quad \sum_{b\in\beta'}\sum_{k=1}^{n}\sum_{t=1}^{T}\sum_{i\in b} a_{i,b}(t+R_{i,b})y_{b,k,t} \tag{4.10}$$

$$S_{k'} \geq \sum_{k=1}^{l}\sum_{b\in\beta'}\sum_{t=1}^{T} P_{b,1}y_{b,k,t} + \sum_{k=l}^{k'-1}\sum_{b\in\beta'}\sum_{t=1}^{T} P_{b,2}y_{b,k,t},$$

$$\forall k' \in \{1,...,n\}, \qquad \forall l \in \{1,...,k'\} \tag{4.11}$$

$$\sum_{b\in\beta'}\sum_{t=1}^{T} y_{b,k,t} \leq 1, \qquad \forall k \in \{1,...,n\} \tag{4.12}$$

$$\sum_{b\in\beta'}\sum_{t=1}^{T}(a_{i,b}\sum_{k=1}^{n} y_{b,k,t}) \geq 1, \qquad \forall i \in \{1,...,n\} \tag{4.13}$$

$$\sum_{b\in\beta'}\sum_{t=1}^{T} t y_{b,k,t} \geq S_k, \qquad \forall k \in \{1,...,n\} \tag{4.14}$$

$$\sum_{b\in\beta'}\sum_{t=1}^{T} y_{b,k,t} \geq \sum_{b\in\beta'}\sum_{t=1}^{T} y_{b,k+1,t}, \qquad \forall k \in \{1,...,n-1\} \tag{4.15}$$

$$\sum_{b\in\beta'}\sum_{t=1}^{T} y_{b,k,t} = 1, \qquad \forall k \in \{1,...,\delta\} \tag{4.16}$$

$$y_{b,k,t} \in \{0,1\} \quad \forall b \in \beta', \forall k \in \{1,...,n\}, \forall t \in \{1,\ldots,T\} \tag{4.17}$$

La fonction objectif (4.10) minimise la somme des dates de livraison des tâches où la date de livraison d'une tâche J_i est donnée par la somme de la date de départ du véhicule pour livrer le lot qui la contient et le temps de transport entre le dépôt et le site i de livraison de j_i. Ainsi D_i est équivalent à $a_{i,b}(t+R_{i,b})y_{b,k,t}$ si la tâche J_i est dans le lot b, la tournée du lot b commence à la date t et si le lot b en position k qui part à la date t est sélectionné. Les contraintes (4.11) garantissent que la production d'un lot commence après celle du précédent et que la date de départ de livraison d'un lot commence après la fin du précédent et après la fin de production du lot transporté. La date de départ d'un lot étant indispensable dans cette formulation, ces contraintes bornent ces dernières et font en sorte que celles-ci soient réalisables. Les contraintes (4.14) garantissent la faisabilité des dates de départ des tournées. Les contraintes restantes (4.12), (4.13), (4.15), (4.16) sont similaires à celles utilisées dans la formulation du problème $1 \to D, k \geq 1 | v = 1, c | D_{max}$.

La génération de colonnes implique que le problème maître représenté par les contraintes (4.10-4.16) soit au préalable restreint à un sous-ensemble de colonnes notées $\tilde{\beta}' \subseteq \beta'$. Ainsi, un sous-problème est nécessaire afin de trouver des éléments de $\beta' \setminus \tilde{\beta}'$ pouvant améliorer la solution de la relaxation continue du problème maître restreint.

Sous-problème (pricing problem) Les variables duales associées aux contraintes (4.11), (4.12),(4.13),(4.14),(4.15) et (4.16) sont respectivement $\alpha'_{lk'}, \beta'_k$, γ'_i, δ'_k, σ'_k et ξ'_k où la variable ξ_k existe uniquement si $k \leq \delta$ et la variable σ_{k-1} si $k \geq 2$ et σ_k si $k \leq n-1$. Le coût

réduit d'une variable y_{bkt} est noté $\bar{c'_{bkt}}$. Le problème $1 \to D, k \geq 1 | v = 1, c | \sum_j D_j$ étant un problème de minimisation, une colonne y_{bkt} peut alors être introduite au problème maître restreint si $\bar{c'_{bkt}} < 0$.

$$\begin{aligned}
\bar{c'_{bkt}} &= \sum_{i=1}^{n}(t + R_{i,b})a_{i,b} + \sum_{k'=1}^{n}\sum_{l=k}^{k'} P_{b,1}\alpha_{l,k'} - \gamma_i a_{i,b} + \sum_{k'=k+1}^{n}\sum_{l=1}^{k} P_{b,2}\alpha_{l,k'} \\
&\quad - \beta_k - t\delta_k + \sigma_{k-1} - \sigma_k - \xi_k, \quad (P_{b,1} = \sum_{i=1}^{n} p_i a_{i,b}) \\
&= \sum_{i=1}^{n}(t + R_{i,b})a_{i,b} + \sum_{i=1}^{n}(p_{i,1}(\sum_{k'=1}^{n}\sum_{l=k}^{k'}\alpha_{l,k'}) - \gamma_i)a_{i,b} + \\
&+ P_{b,2}(\sum_{k'=k+1}^{n}\sum_{l=1}^{k}\alpha_{l,k'}) - \beta_k - t\delta_k + \sigma_{k-1} - \sigma_k - \xi_k
\end{aligned}$$

donc

$$\begin{aligned}
\bar{c'_{bkt}} < 0 \Leftrightarrow \sum_{i=1}^{n}\underbrace{(t + R_{i,b})}_{h_i} a_{i,b} + \sum_{i=1}^{n}\underbrace{(p_{i,1}(\sum_{k'=1}^{n}\sum_{l=k}^{k'}\alpha_{l,k'}) - \gamma_i)}_{l'_i} a_{i,b} + \\
+P_{b,2}\underbrace{(\sum_{k'=k+1}^{n}\sum_{l=1}^{k}\alpha_{l,k'})}_{q} \leq \underbrace{\beta_k + t\delta_k - \sigma_{k-1} + \sigma_k + \xi_k}_{r'_{kt}} \qquad (4.18)
\end{aligned}$$

De même que pour le sous-problème défini pour le problème $1 \to D, k \geq 1 | v = 1, c | D_{max}$, trouver un coût réduit négatif pour une variable y_{bkt} dans le problème $1 \to D, k \geq 1 | v = 1, c | \sum_j D_j$ revient à résoudre un problème de plus court chemin à contraintes de ressources. Afin de représenter le sous-problème, nous utilisons un graphe auxiliaire G' dont seule la longueur des arcs diffère de celles du graphe G introduit dans la section précédente. Nous revenons sur les notations utilisées dans l'expression du coût réduit (4.18)

$$\begin{aligned}
h_i &= t + R_{i,b}, && \forall i \in \{1, ..., n\} \\
l'_i &= p_{i,1}(\sum_{k'=1}^{n}\sum_{l=k}^{k'}\alpha_{l,k'} - \gamma_i), && \forall i \in \{1, ..., n\} \\
q &= \sum_{k'=k+1}^{n}\sum_{l=1}^{k}\alpha_{l,k'}, \\
r'_{kt} &= \beta_k + t\delta_k - \sigma_{k-1} + \sigma_k + \xi_k, && \forall k \in \{1, ..., n\}, \forall t \in T
\end{aligned}$$

Le coût réduit peut s'écrire de la façon suivante.

$$\bar{c'_{bkt}} < 0 \Leftrightarrow \sum_{i=1}^{n} d_i a_{i,b} + \sum_{i=1}^{n} l'_i a_{i,b} + qP_{b,2} < r'_{kt} \qquad (4.19)$$

En raison de la structure de l'expression du coût réduit, plusieurs labels sont nécessaires dans l'algorithme ESPPRC afin de retourner le plus court chemin. En effet, trouver le lot dont la valeur du plus court chemin est minimum consiste à trouver la meilleure combinaison des tâches afin de former un lot. L'équation suivante est minimisée.

$$\min_{b\in\beta'\setminus\tilde{\beta}'}\{\sum_{i=1}^{n} d_i a_{i,b} + \sum_{i=1}^{n} l_i^{'} a_{i,b} + qP_{b,2}\}$$

On peut voir que cette équation est constituée des trois sous-problèmes suivants.

- $\sum_{i=1}^{n} d_i a_{i,b}$: trouver le chemin dont la somme des dates de livraison des tâches est minimale,
- $\sum_{i=1}^{n} l_i^{'} a_{i,b}$: sélectionner des tâches de façon à ce que cette fonction soit minimisée,
- $qP_{b,2}$: trouver le chemin de distance réelle minimale.

Enfin, une colonne y_{bkt} peut être introduite au problème maître restreint si son coût réduit est strictement négatif, i.e,. si la longueur du plus court chemin correspondant à la colonne y_{bkt} est strictement inférieur à $r_{kt}^{'}$. La méthode exacte proposée par Lozano et al est appelée pour chaque position k et date de départ t pour le calcul du plus court chemin (voir [Lozano *et al.*, 2015]). L'initialisation du processus étant similaire à celle utilisée dans la section précédente sur le problème $1 \rightarrow D, k \geq 1|v = 1, c|D_{max}$, nous invitons le lecteur à se référer à l'algorithme utilisé.

4.4 Expérimentations

Afin de valider les approches proposées, nous présentons dans cette section les résultats des tests expérimentaux effectués sur les formulations étendues du problème intégré d'ordonnancement et de tournées de véhicule pour chacune des deux fonctions objectifs. Les résultats présentés dans cette section sont obtenus sur des instances générées aléatoirement de la manière suivante.
Les durées d'exécution et les tailles des tâches sont générées sur des ensembles discrets de distribution uniforme :

$$\begin{aligned} p_j &\in \mathcal{U}(1,100), && \forall j \in \{1,...,n\} \\ s_j &\in \mathcal{U}(1,10), && \forall j \in \{1,...,n\} \end{aligned}$$

Les coordonnés (X_j, Y_j) des sites de livraisons sont générés aléatoirement dans l'intervalle $[1, 40]$ pour chaque site $j \in \{1, \ldots, n\}$ et celles du dépôt sont initialisées à $(0, 0)$. Les distances euclidiennes entre les différents sites sont alors obtenues.

$$\begin{aligned} t_{i,j} = t_{j,i} &= \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}, && \forall i \in \{1,...,n\}, \forall j \in \{1,...,n\} \\ t_{M,j} = t_{j,M} &= \sqrt{(X_J)^2 + (Y_j)^2}, && \forall j \in \{1,...,n\} \end{aligned}$$

Enfin, la capacité du véhicule est fixée à $c = 20$. Etant donné que le problème considéré comporte deux sous-problèmes, i.e., ordonnancement et tournées de véhicule, les données sont générées de sorte à ce que l'un ne prédomine pas sur l'autre. En effet, des durées de transport relativement grandes par rapport aux durées d'exécution des tâches rendent

le problème de tournées de véhicule prédominant au point où la constitution des lots ne dépend que des durées de transport. De même si les durées d'exécution étaient grandes par rapport à celles du transport où un cas extrême serait de mettre $\min_{j\in\{1,\dots,n\}}\{p_j\} \geq \{\max_{i\in\{1,\dots,n\},j\in\{1,\dots,n\}}\{t_{ij}\}, \max_{j\in\{1,\dots,n\}}\{t_{Mj}\}\}$. Dans ce cas, les tâches sont livrées une à une et le problème reviendrait à résoudre un problème d'ordonnancement sur l'objectif $\sum_j D_j$ et à trier les tâches par ordre décroissant des t_{Mj} pour l'objectif D_{max}. Ainsi, les données sont générées afin de constituer un compromis entre les deux sous-problèmes.

Les expérimentations sont implémentées sur une machine Xeon 3.20GHz avec 8GB utilisant ILOG CPLEX 12.6 pour résoudre les programmes linéaires et les algorithmes de plus court chemin sous contraintes de ressources ont été programmés en C++. Nous évaluons et comparons les résultats obtenus par le processus de génération de colonnes sur les problèmes relaxés avec les solutions entières obtenues à partir des colonnes générées par la phase de génération de colonnes. Pour chacun des objectifs, des résultats numériques sont présentés ainsi que la sémantique des paramètres utilisés dans l'algorithme ESPPRC et leurs valeurs. Les résultats sur le problème $1 \to D, k \geq 1|v = 1, c|D_{max}$ sont présentés dans un premier temps avant de passer à ceux du problème $1 \to D, k \geq 1|v = 1, c|\sum_j D_j$.

4.4.1 Résultats sur le problème $1 \to D, k \geq 1|v = 1, c|D_{max}$

Nous commençons par présenter l'algorithme utilisé afin de résoudre le sous-problème de la procédure de génération de colonnes. Ainsi, l'algorithme de plus court chemin élémentaire avec contraintes de ressources proposé par Lozano et al [Lozano *et al.*, 2015] est un algorithme de parcours en profondeur utilisant des étiquettes (labels). Les auteurs font appel à des stratégies qui consistent à détecter les solution irréalisables (noté Infeasibility Prunning) et trouver des bornes supérieures au plus court chemin (notés Bound Pruning). Le parcours en profondeur qui fait qu'une seule étiquette est nécessaire pour chaque noeud ainsi que les règles de dominances utilisées par les auteurs font que cet algorithme est l'un des meilleurs de la littérature. Ainsi, afin de trouver la colonne de coût réduit minimum à rajouter au problème maître restreint, l'algorithme de Lozano est utilisé pour résoudre un ESPPRC dans lequel les ressources sont la taille du lot, i.e., capacité du véhicule et où le calcul des bornes consiste à calculer un plus court chemin de chaque noeud vers la destination en considérant une capacité du véhicule $c' = c - \delta$ avec $\delta = 3i$ et $i \in \{1, \dots, (c-2)/3\}$.

Les solutions obtenues par la génération de colonnes, i.e., solutions de la relaxation continue du problème maître, sont comparées aux solutions entières obtenues par une procédure de branch & bound (B&B) sur les colonnes générées. Ainsi, une solution obtenue par la procédure de génération de colonnes représente une borne inférieure du problème et elle est notée LB. La borne supérieure notée UB est représentée par la solution entière obtenue par le B& B sur les colonnes générées. Dans le but d'obtenir une borne supérieure au problème $1 \to D, k \geq 1|v = 1, c|D_{max}$ dans un temps raisonnable, le sous-problème de la procédure de génération de colonnes introduit une seule colonne à chaque itération. Cependant, afin d'obtenir une bonne borne supérieure, toutes les colonnes de coût réduit négatif sont introduites dans le problème maître restreint à chaque itération. A noter que le fait d'introduire une seule colonne lorsque l'on calcule la borne inférieure a l'avantage de donner rapidement des solutions au problème relaxé du fait de la taille réduite des variables. De même, le fait d'introduire toutes les colonnes de coût réduit négatif a l'avantage de

donner de bonnes bornes supérieures grâce au grand nombre de colonnes générées mais a l'inconvénient d'augmenter le temps d'exécution.

Dans le tableau (4.1), nous nous intéressons aux résultats agrégés sur chaque valeur de n. Les statistiques tiennent compte du temps CPU moyen nécessaire pour arriver à la solution optimale de la relaxation continue (colonne $LB(sec)$). Le nombre de colonnes générées afin d'arriver à la solution optimale de la relaxation continue est donné par la colonne $\#col$. Le gap entre la solution initiale et la solution relaxée est donné par la colonne ($\%GAP_{init}$). Enfin le gap entre la borne supérieure UB obtenue par un B&B sur les colonnes générées et la borne inférieure LB qui représente la solution optimale de la relaxation continue est donné par la colonne ($\%GAP_{UB}$). Les résultats du tableau 4.1 montrent un gap inférieur à 1% pour des instances de taille $n \geq 40$, et un temps d'exécution raisonnable pour atteindre la solution optimale du problème relâché ce qui prouve la qualité des bornes inférieurs obtenues par la procédure de génération de colonnes. A noter qu'un gap réduit sur l'objectif $D_{\max}$ est également du à la nature du problème.

n	$LB(sec)$	$\#col$	$\%GAP_{init}$	$\%GAP_{UB}$
20	1.1	203	21.14	4.62
30	4.6	363	13.67	1.83
40	12.1	595	9.94	0.99
50	46.4	954	10.34	0.88
60	86.4	1136	10.21	0.69
70	215.1	1670	7.30	0.57
80	383.7	1963	7.67	0.42
90	804.8	2637	5.44	0.39
100	1181.9	3041	5.06	–

TABLE 4.1 – Résultats expérimentaux sur le problème $1 \rightarrow D, k \geq 1 | v = 1, c | D_{max}$

4.4.2 Résultats sur le problème $1 \rightarrow D, k \geq 1 | v = 1, c | \sum D_j$

Nous présentons dans cette section les résultats obtenus lorsque l'objectif $\sum D_j$ est considéré. Les notations ainsi que le sous-problème utilisés dans la procédure de génération de colonnes sont identiques à ceux utilisés dans la section précédente pour le problème $1 \rightarrow D, k \geq 1 | v = 1, c | D_{max}$. Cependant, l'expression du coût réduit d'une variable dans le problème $1 \rightarrow D, k \geq 1 | v = 1, c | \sum D_j$ nécessite des étiquettes supplémentaires afin d'obtenir le coût d'un chemin. Nous avons pu voir dans la section précédente que la procédure de génération de colonnes fournit de bonnes solutions lorsque l'objectif $D_{\max}$ est considéré. Cependant, celle-ci est moins efficace sur l'objectif $\sum D_j$ du fait que la convergence est plus lente sur le problème $1 \rightarrow D, k \geq 1 | v = 1, c | \sum_j D_j$ en raison des variables y_{bkt} indexées sur le temps. Les résultats de la méthodes sont donnés par le tableau (4.2) suivant où des écarts important sont observés.

n	$LB(sec)$	$\#col$	$\%GAP_{UB}$
10	17.55	4966	38.1
15	282.39	13513	49.1
20	1811.94	25699	67.99

TABLE 4.2 – Résultats expérimentaux sur le problème $1 \to D, k \geq 1 | v = 1, c | \sum D_j$

4.5 Conclusion

Nous avons étudié dans ce chapitre le cas général du problème intégré d'ordonnancement et de tournées de véhicules, dans lequel la partie tournées de véhicule à été traitée explicitement. Des cas particuliers ont également été étudiés pour lesquels des résultats de complexité ont été proposés. Le problème général étant une extension du problème de voyageur de commerce à tournées multiples, une méthode efficace de génération de colonnes à été appliquée au problème avec succès sur l'objectif $D_{\max}$ en particulier, mais également pour l'objectif $\sum D_j$, avec toutefois des résultats moins performants.

En perspective de ce travail, nous nous concentrerons sur l'amélioration des méthodes proposées et sur une méthode de Branch&Price afin d'obtenir les solutions optimales des problèmes considérés.

Deuxième partie

Production et distribution intégrées - cas robuste

Dans une situation réelle, des événements inattendus peuvent se produire, faisant ainsi varier les données d'un problème. Afin de pallier ce problème, cette incertitude est introduite dans les données du problème et des solutions robustes peuvent être ainsi recherchées. En raison de la vulnérabilité des solutions optimales face à ces incertitudes, les décideurs peuvent se tourner vers des solutions robustes de qualité inférieure mais moins vulnérables. Plusieurs définitions de la robustesse peuvent être trouvées dans la littérature. Nous nous référons au livre de Kouvelis et Yu [Kouvelis et Yu, 1997], où un chapitre entier est consacré aux problèmes d'ordonnancement robuste. De la même manière que [Kouvelis et Yu, 1997], nous considérons que l'incertitude porte sur toutes les données du problème. L'objectif est de proposer une solution qui puisse nous prémunir contre les différentes éventualités qui peuvent surgir. Une telle approche est appelée une *approche robuste*. Une approche basée sur les scénarios est utilisée pour modéliser l'incertitude des données. Plusieurs scénarios sont définis et chacun d'eux correspond à un ensemble de données qui peut potentiellement être réalisé. Le but d'une telle approche est de produire une solution avec une valeur de l'objectif raisonnable quelque soit le scénario réalisé. Dans cette deuxième partie de la thèse, nous proposons une approche originale d'optimisation robuste avec récupération pour un problème intégré d'ordonnancement et de transport. Les travaux réalisés dans cette partie ont été publiés dans [Cheref *et al.*, 2014], [Artigues *et al.*, 2016], [Cheref *et al.*, 2016b] et soumis à une revue internationale [Cheref *et al.*, 2016c].

Chapitre 5

Introduction aux problèmes intégrés d'ordonnancement et de tournées de véhicules robustes

Dans ce chapitre, on se place dans un environnement incertain dans le sens où les données du problème sont soumises à des incertitudes. Dans une première section, nous effectuons un état de l'art des problèmes d'ordonnancement robuste et des problèmes de tournées de véhicules robustes, sachant qu'à notre connaissance il n'existe pas d'approche robuste intégrant les deux problèmes. Dans le section 5.2, nous présentons la structure des groupes de tâches permutables, qui servira de base à la méthode d'optimisation robuste avec récupération que nous proposerons pour résoudre le problème intégré présenté dans la section 5.3.

Une première partie est consacrée à une description synthétique de la robustesse dans laquelle nous examinerons différentes représentations pour le problème d'ordonnancement basées sur une séquence jobs. Nous présentons ensuite la notion de "groupe de tâches permutables" ainsi que son application au problème d'ordonnancement à une machine.

5.1 Revue de la littérature

Nous divisons cet état de l'art en deux parties distinctes. La première partie (Sect. 5.1.1) est consacrée au problème d'ordonnancement robuste et la seconde partie au problèmes de tournées de véhicules robuste (Sect. 5.1.2).

5.1.1 Ordonnancement robuste

"*L'incertitude affecte un large éventail de décisions que les gestionnaires, les ingénieurs et les autres décideurs doivent prendre*" [Kouvelis et Yu, 1997]. Cela est particulièrement vrai pour les gestionnaires de production où les données telles que les durées de traitement des tâches ne sont généralement pas connues avec une certitude absolue. Ceci est également vrai pour les problèmes de livraison où les durées de transport dépendent du temps et sont

sujettes aux fluctuations du trafic. Dans ce contexte, il est généralement important de trouver une solution au problème qui ne soit pas seulement une bonne solution mais aussi une solution qui garantit une bonne performance en présence d'incertitudes. C'est le champ des *approches robustes*. Il existe plusieurs façons pour comprendre et structurer l'incertitude et plusieurs façons de définir une approche robuste [Billaut *et al.*, 2008].

Les problèmes d'ordonnancement robustes sont fréquemment traités dans la littérature et il n'est pas possible de faire ici un l'état de l'art exhaustif sur ce sujet. Nous limitons volontairement notre travail aux approches d'ordonnancement robustes dans lesquelles aucune distribution de probabilité n'est disponible pour les données incertaines. Au lieu de cela, nous considérons uniquement les travaux dans lesquels l'incertitude est représentée sous forme de scénarios sur les données du problème. Cette représentation de l'incertitude pour les problèmes d'ordonnancement est apparue pour la première fois dans [Daniels et Kouvelis, 1995a] (en 1992 dans un rapport de travail [Daniels et Kouvelis, 1995b]). Les auteurs considèrent un environnement à une machine avec des durées de traitement des tâches incertaines où l'incertitude est représentée sous forme d'intervalles, ce qui induit un nombre infini de scénarios. Les auteurs considèrent le temps total du projet comme fonction objectif déterministe et la robustesse est basée sur des écarts absolus ou relatifs par rapport à une performance optimale. Depuis cet article, un nombre conséquent d'études traitent le problème d'ordonnancement sous incertitudes. Nous pouvons citer dans un premier temps deux revues de la littérature qui traitent de l'ordonnancement de projet robuste avec contraintes de ressources [Herroelen et Leus, 2004, Herroelen et Leus, 2005]. Aytug et al. [Aytug *et al.*, 2005] passent en revue la littérature sur les problèmes d'ordonnancement en présence d'événements imprévus et de perturbations et proposent une classification de l'incertitude pour les problèmes d'ordonnancement. Un état de l'art plus récent sur les différentes approches en ordonnancement robuste est proposé par [Verderame *et al.*, 2010].

Dans notre approche, nous intégrons trois familles d'approches pour l'ordonnancement sous incertitudes. La première famille est basée sur la considération implicite de l'incertitude via le concept de calcul d'un ensemble de solutions plutôt qu'une seule solution pour un problème d'ordonnancement dans lequel les données sont sujettes à des perturbations non modélisées. En ordonnancement, une famille de méthodes décrites notamment dans [Billaut et Roubellat, 1996, Artigues *et al.*, 2005a] visent à calculer une séquence de groupes de tâches permutables sur chaque machine de sorte que toute permutation de tâches à l'intérieur d'un groupe donne une séquence de tâches réalisables et garantit une valeur de la fonction objectif respectant une borne supérieure prédéfinie. En d'autres termes, une séquence de groupes de tâches permutables est une solution partielle structurée de sorte que chaque solution complète issue de celle-ci a une performance garantie. L'inconvénient de ces approches est qu'il n'y a aucune indication sur la façon de sélectionner la séquence des tâches à l'intérieur de chaque groupe en réponse à une perturbation particulière. Cependant, dans [Wu *et al.*, 1999], les auteurs ont montré à travers la simulation de diverses perturbations sur les durées d'exécution que le calcul de manière statique d'une telle structure de solution [1], en permettant ensuite aux décisions restantes d'être prises dynamiquement via des règles priorités simples donnant de bons résultats.

1. Ils ont appelé cette structure une "séquence ordonnée de sous-ensembles", ce qui équivaut à la "séquence de groupes de tâches permutables" définie dans [Billaut et Roubellat, 1996]

La seconde famille d'approches est basée sur une optimisation discrète robuste pour l'ordonnancement comme proposé dans [Kouvelis et Yu, 1997]. Dans ce cadre (voir aussi plusieurs autres articles ou ouvrages sur l'optimisation discrète robuste [Bertsimas et Sim, 2003, Bertsimas et Sim, 2004, Aissi *et al.*, 2009, Gabrel *et al.*, 2014]), une séquence complète de production des tâches sur chaque machine est donnée en sortie de la méthode d'ordonnancement robuste. L'incertitude est représentée par un ensemble continu ou discret de scénarios, où chaque scénario correspond à une valeur déterministe des paramètres du problème. Sur un scénario particulier, la performance de la séquence est généralement mesurée en calculant les dates de début au plus tôt des tâches compatibles avec la séquence prescrite et le scénario réalisé. Par conséquent, au moins pour un ensemble de scénarios finis, la performance du pire cas d'une séquence sur l'ensemble de scénarios peut être calculée a priori. A noter que la plupart des problèmes d'ordonnancement robustes (visant à trouver la séquence optimale ayant les meilleures performances au pire cas) sont NP-difficiles y compris pour les problèmes d'ordonnancement à une machine [Aloulou et Della Croce, 2008].

La troisième famille d'approches est basée sur la robustesse avec récupération (recoverable robustness). Cette notion définie par [Liebchen *et al.*, 2009] vise à réduire la conservation (conservativeness) de l'optimisation robuste comme décrite dans [Bertsimas et Sim, 2004]). La robustesse récupérable considère deux ensembles de variables de décision et consiste à trouver une affectation pour les variables de décision du premier niveau ainsi qu'une famille d'algorithmes de réparation de telle sorte que pour chaque scénario, il existe un algorithme de réparation qui calcule des affectations réalisables pour les variables de décision du second niveau. Grâce à cet algorithme, la solution du premier niveau s'adapte au scénario réalisé. A noter que l'ordonnancement robuste standard (deuxième famille de robustesse) peut être considérée comme étant de la robustesse avec récupération. En effet, la solution de sortie d'une solution robuste standard est constituée à la fois d'une séquence de production des tâches (qui peut être considéré comme une affectation des variables de décision du premier niveau) et d'un algorithme qui calcule les dates de début au plus tôt des tâches compatibles avec la séquence prescrite et le scénario réalisé (qui peut être considérée comme un algorithme de second niveau qui affecte des dates de début aux tâches). Récemment, la robustesse avec récupération a été appliquée avec succès à des problèmes tels que l'ordonnancement d'événements avec propagation de retards [Caprara *et al.*, 2014].

Nous définissons maintenant formellement la notion de robustesse avec récupération en ligne (online recoverable robustness). Considérons d'abord le problème d'optimisation robuste que nous pouvons noter de la manière suivante

$$\min_{x \in \cap_{s \in \mathcal{S}} \mathcal{X}_s} \max_{s \in \mathcal{S}} f(x, s)$$

où $\mathcal{S}$ indique l'ensemble des scénarios, $\mathcal{X}_s$ l'ensemble des solutions réalisables sur un scénario s et, $f(x, s)$ un indicateur de performance d'une solution $x \in \mathcal{X}_s$ sur un scénario $s \in \mathcal{S}$. Cette définition implique que x doit être réalisable pour chacun des scénarios de sorte que le degré de pessimisme induit soit important [Bertsimas et Sim, 2004]. Le concept de robustesse recouvrable (recoverable robustness) [Liebchen *et al.*, 2009] est également considéré dans la littérature comme étant une extension de la robustesse standard consistant à traiter le problème en deux étapes. Ainsi, des modification limitées sont apportées à une solution x obtenue en première étape de sorte à s'adapter au scénario réalisé et obtenir une nouvelle solution $y = A(x, s)$ où, $A(x, s)$ est un algorithme de récupération (*application-dependent*

recovery algorithm) d'une solution x sur un scénario s. Dès lors, le problème d'optimisation robuste devient recouvrable :

$$\min_{x \in \mathcal{X}} \max_{s \in \mathcal{S}} f\left(A(x,s), s\right)$$

où $\mathcal{X}$ est une solution obtenue durant la première étape ne dépendant pas des scénarios. A savoir que x n'affecte pas nécessairement toutes les variables de décision du problème. De la même manière que la programmation stochastique, x peut représenter uniquement un sous-ensemble des variables de décision de la première étape alors que l'algorithme $A(x,s)$ retourne une solution complète. Cette dernière solution retourne les valeur des variables de décision de la première étape ainsi que celles de la deuxième étape.

5.1.2 Tournées de véhicules robustes

La robustesse à également été considérée dans les problèmes de tournées de véhicules et ce depuis les année 1990. Bertsimas et Simchi-Levi dans [Bertsimas et Simchi-Levi, 1996] présentent une revue de littérature des problèmes de tournées de véhicules sous incertitudes. Dans la plupart des problèmes traités dans la littérature, les auteurs considèrent des demandes stochastiques en considérant que celles ci arrivent aléatoirement ou que la quantité demandée est aléatoire, etc. Ils proposent des algorithmes robustes indépendants de la variabilité des données. La même année, dans [Gendreau *et al.*, 1996] les auteurs proposent une revue de littérature sur les problèmes de tournées de véhicules stochastiques où sont répertoriés les travaux incluant des demandes stochastiques, des temps de transport stochastiques et des clients stochastiques. En plus de cette littérature sur les tournées de véhicules basée sur des approches de programmation stochastique, des méthodes d'optimisation robustes ont été abordées plus récemment. Un état de l'art sur l'optimisation robuste pour le problème de tournées de véhicules avec fenêtres de temps à été proposé dans [Agra *et al.*, 2013]. Le même article traite le problème dans lequel l'incertitude sur les temps de transport est représentée sous forme de polytope. Parallèlement à l'optimisation robuste standard, [Ben-Tal *et al.*, 2004] considèrent un contexte de robustesse ajustable proche de la structure de robustesse avec récupération à l'exception que dans leurs travaux, les variables de second niveau peuvent être ajustées par rapport au scénario réalisé sans restriction particulière sur l'algorithme (alors qu'un ensemble d'algorithmes prédéfini est utilisé dans la robustesse récupérable). D'autres auteurs ont considéré une optimisation robuste pour un problème de tournées de véhicules avec capacités avec une demande incertaine [Gounaris *et al.*, 2013].

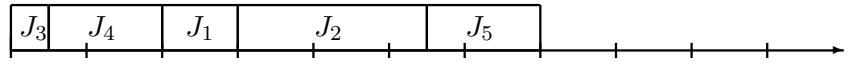
5.1.3 Exemple de robustesse en ordonnancement

L'exemple suivant illustre le concept de robustesse sur le problème d'ordonnancement à une machine. Soient r_j, p_j et d_j la date de début au plus tôt, la durée d'exécution et la date de fin souhaitée d'une tâche J_j pour un scénario s. La Table 5.1 présente les données du problème et la figure 5.1 illustre les séquences σ_{ERD} (tri par ordre croissant sur les dates de début au plus tôt) et σ_{EDD} (tri par ordre croissant sur les dates de fin) ainsi que leurs résultats respectifs sur le deuxième scénario.

	J_1	J_2	J_3	J_4	J_5
r_j	3	3	0	1	7
p_j	2	5	1	3	3
d_j	6	11	2	5	14

TABLE 5.1 – Instance d'un problème d'ordonnancement à une machine

Nous pouvons voir que σ_{ERD} et σ_{EDD} retournent une même séquence $(J_3, J_4, J_1, J_2, J_5)$. Soit maintenant un second scénario dont les données sont présentées dans la Table 5.2.



$$L_{\max}(\sigma_{ERD}) = L_{\max}(\sigma_{EDD}) = \max(0, 0, -1, -1, 0) = 0$$

FIGURE 5.1 – Représentation des séquences σ_{ERD} et σ_{EDD} et leurs évaluations.

	J_1	J_2	J_3	J_4	J_5
r_j	0	7	3	4	3
p_j	3	4	1	2	4
d_j	3	14	4	6	10

TABLE 5.2 – Instance d'un problème d'ordonnancement à une machine sur le second scénario

Si on compare les solutions σ_{ERD} et σ_{EDD} retournées sur ce deuxième scénario à celle retournées sur le premier scénario, on peut dire que σ_{EDD} est plus robuste que σ_{ERD} car elle retourne une solution du pire cas égale à 0 alors que la séquence σ_{ERD} retourne une solution du pire cas égale à 14.

5.2 Groupes de tâches permutables : une structure de solution pour l'ordonnancement robuste

Une séquence de groupes de tâches permutables sur une machine M_k est une partition ordonnée de l'ensemble des tâches $\mathcal{J}$ sur la machine M_k. Un élément de chaque partition est appelé "groupe de tâches permutables". De la même manière que pour une séquence de tâches, le terme "séquence de groupes" est utilisé pour dénoter une séquence d'un ensemble de groupes sur machine. Cette structure à été proposée en 1990 par François Roubellat [Lopez et Roubellat, 2008]. Nous présentons dans cette section la notion de groupes de tâches permutables plus en détails et passons en revue les différents modèles et algorithmes proposés dans la littérature afin d'évaluer la séquence de groupes.

5.2.1 Exemple : Environnement à une machine

Soit une séquence de groupes sur le second scénario de l'exemple précédent. Cette séquence est constituée par un premier groupe composé des tâches $\{J_1, J_3\}$ et d'un deuxième groupe composé des tâches $\{J_2, J_4, J_5\}$ (voir la figure 5.2). Ainsi, la production du premier groupe commence à la date 3 car J_3 ne peut pas commencer avant la date 3. La durée d'un groupe est équivalente à la durée de production des tâches qui le composent. Nous pouvons voir sur cet exemple que quelque soit l'ordre de production des tâches à l'intérieur du premier groupe, la production du second groupe peut commencer à la date 7 (à une date antérieure si la tâche J_1 est produite en premier). Nous pouvons voir également que la flexibilité apportée par la séquence de groupes (12 séquences sont caractérisées) a un prix étant donné que la date de fin de production totale est égale à 17 au lieu de 14 dans l'exemple précédent.

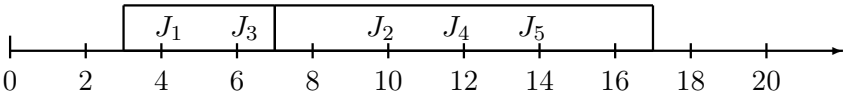


FIGURE 5.2 – Représentation d'une séquence de groupes sur un problème d'ordonnancement à une machine

5.2.2 Problèmes d'optimisation combinatoire sur la séquence de groupes

Plusieurs problèmes d'optimisation combinatoire peuvent être définis sur une solution représentée sous forme de séquence de groupes et ont été étudiés dans la littérature. Nous nous restreignons aux problèmes d'ordonnancement où la production des tâches se fait au plus tôt, ce qui est dominant pour les fonctions objectifs régulières en ordonnancement. Étant donné qu'une séquence de groupes G représente en général un nombre exponentiel de solutions (séquences de tâches) réalisables, une question se pose sur la façon de sélectionner une séquence de tâches sur un scénario donné s. Dans un cadre d'optimisation robuste et en particulier de robustesse avec récupération, étant donné un problème d'ordonnancement disjonctif et un ensemble de scénarios, une séquence de groupes G est alors calculée durant la première étape de la décision. Une fois le scénario réalisé révélé, la décision de seconde étape consiste alors à sélectionner une séquence de tâches parmi toutes les solutions possibles à l'aide d'un algorithme $A(G, s)$.

Un exemple typique serait de définir $A(G, s)$ comme étant un algorithme de liste qui sélectionne un ordre à l'intérieur de chaque groupe selon une règle de priorité. En effet, nous pouvons identifier l'ensemble des algorithmes de listes pour l'ordonnancement compatibles avec G ainsi que l'ensemble des séquences de tâches représentées par G. Cela donne lieu à plusieurs problèmes d'optimisation combinatoire. Soient $\mathcal{G}$ l'ensemble des groupes caractérisés, $\sigma(G)$ une séquence d'ordonnancement des tâches du groupe G et $\Sigma(G)$ l'ensemble des séquences de tâches du groupe G.

Meilleur ordonnancement au plus tôt d'une séquence de groupes sur un scénario fixé L'objectif de l'algorithme $A(G, s)$ de seconde phase est naturellement d'obtenir le meilleur ordonnancement suivant le scénario réalisé. Le problème correspondant est appelé (GP_1) et

constitue une borne inférieure pouvant être atteinte par l'algorithme $A(G, s)$.

$$(GP_1) \quad \min_{\sigma \in \Sigma(G)} L^s_{\max}(\sigma)$$

Dans le cas d'un problème d'ordonnancement à une machine où la séquence de groupes est constituée d'un seul groupe (i.e., $|G| = 1$), trouver la meilleure séquence des tâches du groupe revient à résoudre le problème NP-difficile d'ordonnancement à une machine $1|r_i|L_{\max}$ [Lenstra *et al.*, 1977].

Pire ordonnancement au plus tôt d'une séquence de groupes sur un scénario fixé Ce problème noté (GP_2) cherche la pire performance que tout algorithme $A(G, s)$ peut atteindre sur un scénario donné ce qui donne une borne supérieure de la performance des décisions de seconde étape. En conjonction avec le problème (GP_1), nous pouvons obtenir une borne inférieure et une borne supérieure de la performance de tout algorithme de deuxième étape compatible avec la séquence de groupes G et sur un scénario donné.

$$(GP_2) \quad \max_{\sigma \in \Sigma(G)} L^s_{\max}(\sigma)$$

De même que pour le problème précédent, un cas limite est de considérer un seul groupe de n tâches pour le problème d'une machine. Dans ce cas, nous obtenons des problèmes de maximisation d'un ordonnancement à une machine, qui sont en général plus faciles que leur équivalent de minimisation comme le montrent Aloulou et al [Aloulou *et al.*, 2004] qui fournissent des algorithmes polynomiaux et des preuves de complexité pour plusieurs cas. En outre, dans [Artigues *et al.*, 2005b], il a été montré que (GP_2) est polynomial pour tout problème d'ordonnancement disjonctif. Le calcul du pire ordonnancement au plus tôt consiste à calculer pour chaque tâche les pires dates de début de production au plus tôt et dates de fin de production au plus tôt.

Maximisation de flexibilité avec un objectif borné sur un scénario fixé Sans aucune hypothèse sur l'algorithme de la deuxième étape $A(G, s)$, il reste à déterminer une mesure de flexibilité afin d'évaluer la séquence de groupe G. L'objectif est alors de maximiser la flexibilité de la solution (notée $flex(G)$) tout en s'assurant qu'une borne supérieure UB de la fonction objectif ne soit pas dépassée. Par conséquent, ceci revient à proposer une séquence de groupes dans laquelle un maximum de séquences de tâches satisfont la borne supérieure UB. Le problème (GP_3) est alors posée.

$$(GP_3) \quad \max_{G \in \mathcal{G}} flex(G) \text{ s.t. } L^s_{\max}(\sigma) \leq UB, \forall \sigma \in \Sigma(G)$$

Comme résultat du problème (GP_3), nous obtenons une séquence de groupes garantissant une performance sur chaque scénario mais celle-ci ne donne cependant aucune indication sur la meilleure solution pouvant être trouvée. Pour ce faire, le problème (GP_1) doit être résolu. Le problème (GP_1) étant généralement NP-difficile, une alternative serait de demander à ce que dans la séquence de groupes soit représentée une séquence de tâches σ^0. Cette manière de faire donne le problème suivant.

$$(GP_4) \quad \max_{G \in \mathcal{G}} flex(G) \text{ s.t. } \sigma^0 \in \Sigma(G), L^s_{\max}(\sigma) \leq UB, \forall \sigma \in \Sigma(G)$$

Où σ^0 est la séquence optimale sur un scénario s. Ainsi, la résolution de (GP_4) retourne une séquence de groupes G avec un maximum de flexibilité qui vérifie pour toute séquence σ issue de G les bornes suivantes.

$$L^s_{\max}(\sigma^0) \leq L^s_{\max}(\sigma) \leq UB$$

. Nous pouvons trouver dans la littérature plusieurs mesures de flexibilité. Une des plus naturelles est le nombre de séquences de tâches issues d'une séquence de groupes.

$$flex_1(G) = |\Sigma(G)|$$

Dans [Artigues *et al.*, 2005b], les auteurs utilisent le nombre de groupes comme mesure de flexibilité. En effet et de manière intuitive, un nombre de groupes petit peut apporter plus de flexibilité. Cette mesure est définie comme suit.

$$flex_2(G) = |G|$$

Dans [Artigues *et al.*, 2005b], un algorithme en $O(n^3)$ est proposé pour résoudre le problème (GP_3) en utilisant la mesure de flexibilité $flex_1$. Les auteurs traitent un problème d'ordonnancement à une machine sans contrainte sur les dates de début au plus tôt des tâches (i.e., $r_j = 0, \forall j \in \mathcal{J}$) et où la date de fin de production souhaitée d'une tâche dépend de sa durée de production (i.e., $i, j \in \mathcal{J}$, $p_i \leq p_j \Leftrightarrow d_i \leq d_j$). Le même algorithme est appliqué au problème (GP_4) en utilisant la mesure de flexibilité $flex_2$ ainsi que le problème (GP_3) en utilisant $flex_2$. En prenant en compte les dates de début au plus tôt des tâches, le problème (GP_4) est résolu en $O(n^7)$ pour $flex_1$ et en $O(n^4)$ pour $flex_2$, la mesure de flexibilité $flex_2$ étant généralement plus simple à traiter que $flex1$.

Optimisation robuste avec récupération via des algorithmes d'ordonnancement de listes sur l'ensemble des scénarios Nous considérons maintenant un algorithme de liste $A(G, s)$ qui retourne une séquence de tâches qui soit compatible avec la séquence de groupes G et réalisable pour le scénario s. Par conséquent, la séquence de groupes recherchée est celle qui maximise la robustesse sur le scénario réalisé s par l'intermédiaire de l'algorithme de liste $A(G, s)$. Le problème est noté (GP_6) et est comme suit.

$$(GP_6) \quad \min_{G \in \mathcal{G}} \max_{s \in S} L^s_{\max}(A(G, s))$$

A noter que la représentation sous forme de groupes introduit un niveau de décision supplémentaire comparé à la représentation sous forme de séquence de tâches. Ainsi, le premier niveau de décision construit la séquence de groupes, le deuxième sélectionne la séquence des tâches dans chaque groupe et le dernier niveau de décision sélectionne les dates de début des tâches.

Dans le chapitre 7, le problème (GP_6) est résolu pour le problème d'ordonnancement à une machine et est comparé à la version robuste standard.

5.3 Description du problème d'ordonnancement et de tournées de véhicules robuste

Suivant la notation classique des problèmes d'ordonnancement proposée par [Graham *et al.*, 1979], Z-L. Chen a introduit dans [Chen, 2010b] une notation $\alpha|\beta|\pi|\delta|\gamma$

pour les problèmes intégrés d'ordonnancement et de distribution à un niveau opérationnel. Dans ce qui suit, nous nous concentrons sur un environnement à une machine ($\alpha = 1$) avec un problème de tournées de véhicules pour la partie livraison. La livraison est effectuée par un seul véhicule de capacité illimitée ($c = n$), i.e., la capacité du véhicule est suffisante pour transporter l'ensemble des tâches disponibles.

Nous considérons un ensemble $\mathcal{J} = \{J_1, ..., J_n\}$ de n tâches à produire sur le site de production représenté par une seule machine. Pour chaque tâche J_j, en plus de la durée de production p_j, nous introduisons dans ce chapitre une date de début de production au plus tôt noté r_j et une date de livraison souhaitée notée d_j. Concernant la partie distribution, le site de production est noté 0 et à chaque tâche J_j est associée un site de livraison (client) noté j. Le temps de transport entre le site de production 0 et un site j est noté $t_{0,j}$ et le temps de transport entre deux sites i et j est noté $t_{i,j}$ ($\forall i, j, 1 \leq i, j, \leq n$). Pour toute tâche J_j ($1 \leq j \leq n$), sa date de fin de production sur la machine est représentée par la variable C_j et sa date de livraison (date d'arrivée au client j) est représentée par la variable D_j. Nous rappelons que le véhicule peut effectuer plusieurs tournées afin de livrer toutes les tâches et par conséquent retourner à chaque fois au site de production pour transporter de nouvelles tâches. Ainsi, le problème sans incertitude sur les données peut être noté $1|r_j|routing, V(1,\infty)|n|\gamma$ où γ représente la fonction objectif. Dans la suite du chapitre, nous considérons l'objectif à minimiser γ comme étant le retard de livraison maximum $L_{\max}$ défini comme suit.

$$L_{\max} = \max_{1 \leq j \leq n} (D_j - d_j)$$

Le problème consiste alors à trouver une séquence de production de manière à déterminer la date de fin de production des tâches. Une fois les dates de disponibilités des tâches connues, le problème consiste à regrouper les tâches en lots et a déterminer la meilleure route pour chacun des lot de sorte à minimiser la fonction objectif. A noter qu'une tournée commence lorsque toutes les tâches contenues dans le véhicule sont disponibles.

Nous supposons dans cette partie que des incertitudes portent sur les données d'entrée et l'objectif est de trouver une solution robuste. Afin de représenter l'incertitude, nous utilisons une approche basée sur les scénarios similaire à celle introduite dans [Kouvelis et Yu, 1997]. Ainsi, les données relatives aux tâches et les durées de transport varient suivant un ensemble prédéfini de scénarios. Soit $\mathcal{S}$ l'ensemble des scénarios possibles dans un horizon de temps. Nous considérons maintenant que pour chacune des tâches J_j sont associées une dates de début au plus tôt r_j^s, des durées de production p_j^s et des dates de livraisons souhaitées d_j^s pour tout scénario $s \in \mathcal{S}$. Les temps de transport étant affectés par les incertitudes, la matrice $\mathcal{T}^s = (t_{i,j}^s)_{0 \leq i \leq n, 0 \leq j \leq n}$ donne les temps de transport de tout site i vers tout site j pour chacun des scénarios $s \in \mathcal{S}$. De la même manière, la date de fin de production d'une tâche J_j sur un scénario s est notée C_j^s et sa date de livraison notée D_j^s. Ainsi, le retard de livraison d'une tâche J_j sur un scénario s noté L_j^s est égal à $D_j^s - d_j^s$. En prenant en compte l'incertitude, la fonction objectif suivante est introduite.

Pire cas des plus grand retards de livraison : $L_{\max} = \max_{s \in \mathcal{S}}(\max_{1 \leq j \leq n} L_j^s)$.

Afin d'adapter le concept de robustesse recouvrable au problème intégré d'ordonnancement et de tournées de véhicules, nous introduisons les notations suivantes. On appelle π

une séquence complète ou partielle de production des tâches sur la machine, $\mathcal{B}$ l'ensemble des batchs qui constitue une partition des tâches et $\Sigma = \{\sigma(B), B \in \mathcal{B}\}$ une tournée complète ou partielle pour chacun des batchs, .i.e., $\sigma(B)$ retourne une séquence complète ou partielle de livraison des tâches du batch $B \in \mathcal{B}$.

Soient C^s et D^s les vecteurs de la seconde étape donnant respectivement les date de fin de production et de livraison pour un scénario s. Étant donnée une solution $x = (\pi, \mathcal{B}, \sigma)$ retournée par la première étape, l'algorithme de récupération $A(x, s)$ retourne une solution réalisable $y = A(x, s) = (C^s, D^s)$. Conformément à la fonction objectif, l'indicateur de performance $f(y, s)$ est le retard maximum $L_{\max}$. Les problèmes d'ordonnancement ainsi que de tournées de véhicules étant contraints par le temps, nous supposons que l'algorithme $A(x, s)$ est un algorithme *online*, i.e., appelé durant l'exécution du problème. A un instant de décision t et avec comme seules connaissances le scénario révélé à t (les tâches restant à produire et à livrer à la date t sont connues), l'algorithme doit rapidement retourner une décision simple (i.e., à l'instant t, quelle est la prochaine tâche à exécuter sur la machine ou quel est le prochain site à visiter dans une tournée).

Les chapitres 6 et 7 présentent deux définitions différentes des variables de décision $x = (\pi, \mathcal{B}, \sigma)$ de la première étape. Nous présentons d'abord une première version inspirée de [Kouvelis et Yu, 1997] dans laquelle une séquence complète de production et de livraison est retournée. Ensuite, une seconde version qui retourne une séquence partielle est présentée. En d'autres termes, nous introduisons dans la seconde version le concept de groupes de tâches permutables [Billaut et Roubellat, 1996, Artigues *et al.*, 2005a] dans le cadre de la robustesse avec récupération. A ces deux méthodes, nous définissons deux algorithmes online afin de retourner les variables de décision $y = (C^s, D^s)$ de la seconde étape.

5.4 Conclusion

Dans ce chapitre nous avons présenté un état de l'art rapide des approches d'optimisation robuste appliquées aux problèmes d'ordonnancement et de tournées de véhicules. Il apparaît qu'il n'existe aucune proposition d'approches robustes pour le problème intégré d'ordonnancement et de tournées de véhicules. Par ailleurs nous avons présenté la structure de groupes de tâches permutables et nous montrons qu'elle est adaptée pour servir de support aux approches d'optimisation robuste avec recours. Finalement, nous présentons le problème d'ordonnancement et de tournées robuste, qui sera abordé par des méthodes d'optimisation robustes "standard" dans le chapitre 6 et par des méthodes d'optimisation robustes avec récupération dans le chapitre 7.

Chapitre 6

Résolution du cas robuste par l'approche "standard"

Nous présentons dans ce chapitre des modèles mathématiques et une heuristique pour le problème intégré d'ordonnancement et de tournées de véhicules robuste en utilisant le concept de robustesse proposé par [Kouvelis et Yu, 1997]. Le problème consiste à trouver une séquence complète de production π, l'ensemble des batchs $\mathcal{B}$ et pour chacun des batchs $B \in \mathcal{B}$, une séquence de livraison complète. Dans la section 6.1, nous présentons trois modélisations du problème d'ordonnancement robuste dans lesquelles la partie livraison est ignorée. Dans la section 6.2, nous proposons également plusieurs PLNE pour le problème intégré d'ordonnancement et de tournées de véhicules. Dans chacun des cas cités, un exemple illustratif est donné. Enfin dans la section 5.3, nous présentons une métaheuristique de type tabou pour le problème. Des expérimentations numériques sont données au paragraphe 6.3.

6.1 Formulation PLNE du problème d'ordonnancement robuste "standard"

Dans cette section, nous traitons le problème d'ordonnancement avec des dates de début de production au plus tôt r_j ainsi que des dates de fin de production au plus tard d_j. L'objectif est de minimiser le retard de production maximum L_{max} sur tous les scénarios, où $L_{\max} = \max_{s\in\mathcal{S}}(\max_{1\leq j\leq n} L_j^s)$ et $L_j^s = C_j^s - d_j^s, \forall s \in \mathcal{S}$. A noter que le problème est NP-difficile avec un seul scénario [Lenstra *et al.*, 1977]. L'objectif est alors de trouver la meilleure séquence de production possible π pour tous les scénarios. Etant donné π, l'algorithme on-line ajuste les tâches par rapport au scénario réalisé en faisant en sorte de minimiser les dates de fin de production des tâches C_i^s. Le premier modèle noté (SM) que nous proposons est basée sur celui proposé par [Kouvelis et Yu, 1997] pour le problème $1||\sum C_j$.

Nous définissons les variables suivantes pour le modèle.

— $x_{j,k} \in \{0,1\}$ vaut 1 si la tâche J_j est en position k dans la séquence π, 0 sinon.

Minimiser $L_{\max}$

$$\sum_{j=1}^{n} x_{j,k} = 1, \qquad \forall k \in \{1,...,n\} \tag{6.1}$$

$$\sum_{k=1}^{n} x_{j,k} = 1, \qquad \forall j \in \{1,...,n\} \tag{6.2}$$

$$L_{\max} \geq \sum_{j=1}^{n} r_j^s x_{j,k} + \sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q} - \sum_{j=1}^{n} d_j^s x_{j,k'}, \quad \forall k, k' \in \{1,...,n\}, k \geq k', \forall s \in \mathcal{S} \tag{6.3}$$

$$x_{j,k} \in \{0,1\}, \qquad \forall j,k \in \{1,...,n\} \tag{6.4}$$

Les contraintes (6.1) et (6.2) garantissent la faisabilité de la séquence en affectant exactement une tâche à chaque position. Les contraintes (6.3) expriment le fait que le retard maximum dans le pire cas $L_{\max}$ est supérieur ou égal au retard maximum pour chacun des scénarios. Effectivement, $\sum_{j=1}^{n} r_j^s x_{j,k}$ donne la date de début au plus tôt de la tâche en position k pour le scénario s et $\sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q}$ est égal à la durée totale de production des tâches produites entre les positions k et k' sur un scénario s alors que $\sum_{j=1}^{n} d_j^s x_{j,k'}$ donne la date de fin de production au plus tard de la tâche en position k' pour le scénario s. Ces contraintes sont basées sur le fait que dans une solution calée au plus tôt, il existe un bloc de tâches dont la production commence à la date de début au plus tôt de la première tâche du bloc et où les tâches sont produites sans interruption, ce qui donne le plus grand retard. Par conséquent, les variables indiquant les dates de fin de production des tâches ne sont pas nécessaires pour la deuxième phase de l'algorithme.
Ce modèle contient n^2 variables binaires, une variable continue et $O(n^2|\mathcal{S}|)$ contraintes.

Nous proposons dans ce qui suit une deuxième formulation du problème notée (*SM2*). Les variables de décision utilisées dans la formulation (*SM*) sont maintenues et des variables sont introduites pour la seconde phase de l'algorithme.

— $\tilde{C}_k^s > 0$ est la date de fin de production de la tâche en position k pour le scénario $s \in \mathcal{S}$.

Minimiser $L_{\max}$

$$\tilde{C}_k^s \geq \tilde{C}_{k-1}^s + \sum_{j=1}^{n} p_j^s x_{j,k}, \qquad \forall k \in \{2,...,n\}, \forall s \in \mathcal{S} \tag{6.5}$$

$$\tilde{C}_k^s \geq \sum_{j=1}^{n} (r_j^s + p_j^s) x_{j,k}, \qquad \forall k \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.6}$$

$$L_{\max} \geq \tilde{C}_k^s - \sum_{j=1}^{n} d_j^s x_{j,k}, \qquad \forall k \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.7}$$

$$\tilde{C}_k^s \geq 0 \qquad \forall k \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.8}$$

Pour un scénario $s \in \mathcal{S}$, les contraintes (6.5) font en sorte que la date de fin de production d'une tâche en position k soit supérieure ou égale à sa durée de production plus la date de fin de production de la tâche qui la précède. Les contraintes (6.6) expriment le fait que la date de fin de production d'une tâche en position k sur un scénario s est supérieure ou égale à sa date de début au plus tôt plus sa durée de production. La variable $L^s_{\max}$ à minimiser est évaluée par les contraintes (6.7).
Ce modèle contient n^2 variables binaires, $n|\mathcal{S}|$ variables continues et $O(n|\mathcal{S}|)$ contraintes. Nous pouvons voir que cette formulation contient moins de contraintes mais plus de variables que la formulation précédente.

La troisième formulation proposée notée *SM3* est basée sur des variables de précédences définissant la séquence de production. Des variables explicite sont ici utilisées pour retourner les dates de fin de production des tâches.

— $y_{i,j} \in \{0,1\}$ vaut 1 si la production de la tâche J_i précède immédiatement celle de la tâche J_j, 0 sinon.
— $\tilde{C}^s_j > 0$ retourne la date de fin de production de la tâche en position J_j sur le scénario $s \in \mathcal{S}$.

Deux tâches fictives J_0 et J_{n+1} sont également introduites au modèle (il n'y a pas besoin de définir les durées de production et les dates de fin de J_0 et J_{n+1}). Ainsi, une solution du problème constitue un chemin de J_0 à J_{n+1}.

Minimiser $L_{\max}$

$$\sum_{j=1, j\neq i}^{n+1} y_{i,j} = 1, \qquad \forall i \in \{0,...,n\} \tag{6.9}$$

$$\sum_{i=0, i\neq j}^{n} y_{i,j} = 1, \qquad \forall j \in \{1,...,n+1\} \tag{6.10}$$

$$C^s_j \geq C^s_i + p^s_j - M(1 - y_{i,j}), \qquad \forall i \in \{1,...,n\}, \forall j \in \{1,...,n\}, i \neq j, \forall s \in \mathcal{S} \tag{6.11}$$

$$C^s_j \geq r^s_j + p^s_j, \qquad \forall j \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.12}$$

$$L_{\max} \geq C^s_j - d^s_j, \qquad \forall j \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.13}$$

$$y_{i,j} \in \{0,1\} \qquad \forall i \in \{0,...,n\}, \forall j \in \{1,...,n+1\}, i \neq j \tag{6.14}$$

$$C^s_j \geq 0 \qquad \forall i \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.15}$$

Un chemin entre la tâche J_0 et la tâche J_{n+1} traversant toutes les tâches est construit à l'aide des contraintes (6.9) et (6.10). Les sous-tours sont implicitement éliminés grâce aux contraintes (6.11). Les contraintes (6.11) et (6.12) garantissent qu'une tâche commence après la fin de son prédécesseur et après sa date de début de production au plus tôt. A noter que les contraintes (6.11) nécessitent l'utilisation d'une grande constante M et dès lors, la qualité de la résolution peut être compromise.
Ce modèle contient $O(n^2)$ variables binaires, $n|\mathcal{S}| + |\mathcal{S}|$ variables continues et $O(n^2|\mathcal{S}|)$ contraintes.

Exemple

Nous considérons dans cet exemple une instance avec $n = 5$ et un nombre de scénarios $|\mathcal{S}| = 2$. Le tableau suivant présente les données du problème.

$s=1$	J_1	J_2	J_3	J_4	J_5
r_j^1	0	7	3	4	3
p_j^1	3	4	1	2	4
d_j^1	3	14	4	6	10

$s=2$	J_1	J_2	J_3	J_4	J_5
r_j^2	3	3	0	1	7
p_j^2	2	5	1	3	3
d_j^2	6	11	2	5	14

La solution robuste optimale pour ce problème est donnée par la séquence $(J_3, J_1, J_4, J_5, J_2)$ telle que le retard maximum $L_{\max}$ est égal à 5. La figure (Fig. 6.1) illustre les dates de fin de production des tâches ainsi que le retard maximum pour les deux scénarios.

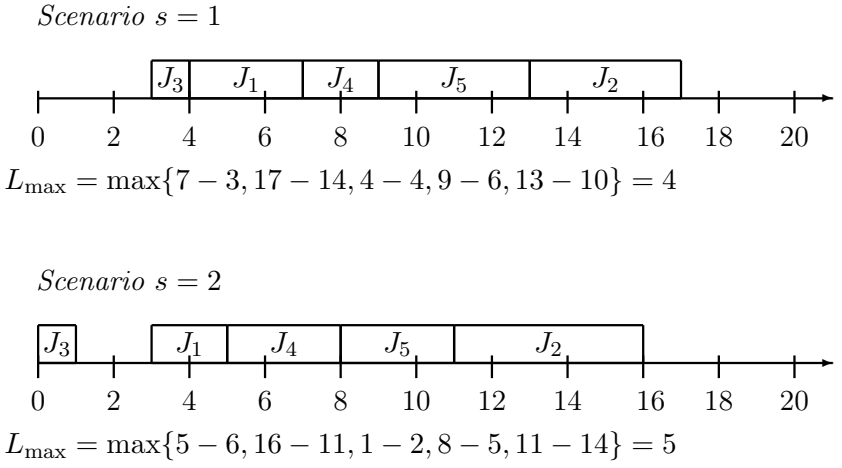


FIGURE 6.1 – Séquence $(J_3, J_1, J_4, J_5, J_2)$ et ordonnancements pour les deux scénarios possibles

6.2 Formulation PLNE du problème intégré d'ordonnancement et de tournées de véhicule robuste standard

Nous présentons dans cette section un nouveau PLNE dans lequel le problème de tournée de véhicules est intégré au problème d'ordonnancement robuste. Les formulations proposées reprennent les modèles d'ordonnancement robuste présentés précédemment. Du fait que la partie livraison soit intégrée au problème, les variables du premier niveau donnent une séquence de production complète π, une partition des tâches en batchs $\mathcal{B}$ ainsi que la séquence de livraison $\sigma(B)$ pour chaque batch $B \in \mathcal{B}$.

Dans les formulations suivantes, nous utilisons les variables binaires suivantes associées à la partie tournée de véhicules du problème. Soit B_r un batch livré à la r^e position et $\sigma(B_r)$ la tournée lui correspondant où r varie de 1 à n et où certains batchs peuvent être vides.

— $z_{i,j,r} \in \{0,1\}$ vaut 1 si la production de la tournée $\sigma(B_r)$ visite consécutivement le site i puis le site j, 0 sinon.
— $y_{j,r} \in \{0,1\}$ vaut 1 si la tournée $\sigma(B_r)$ visite le site j (si la tâche J_j est dans le batch B_r), 0 sinon.
— $D^s_{j,r} > 0$ est une variables de second niveau et retourne la date de livraison pour un scénario $s \in \mathcal{S}$ d'une tâche J_j du batch B_r.

Deux tâches J_0 et J_{n+1} fictives sont introduites au modèle afin d'avoir les dates de départs et d'arrivées des tournées. Ainsi, pour un scénario $s \in \mathcal{S}$, la variable $D^s_{0,r}$ retourne la date de départ de la tournée $\sigma(B_r)$ et la variable $D^s_{n+1,r}$ retourne la date de retour au dépôt de la tournée $\sigma(B_r)$.

Dans la formulation qui suit (noté *SRM1*), la partie tournée de véhicules est introduite au modèle d'ordonnancement robuste SM. Les variables de seconde phase $\tilde{C}^s_k$ étant nécessaires dans ce modèle, celles-ci sont introduites par les contraintes (6.16). En effet, les variables $\tilde{C}^s_k$ sont nécessaires pour obtenir les dates de départ au plus tôt du véhicule. Cette formulation est basée sur le concept de blocs et utilise les contraintes (6.17)-(6.23) afin de déterminer le retard maximum dans le pire cas $L_{\max}$ préalablement obtenue dans SM par les contraintes (6.1).

Minimiser $L_{\max}$

$$\sum_{j=1}^{n} r^s_j x_{j,k} + \sum_{q=k}^{k'} \sum_{j=1}^{n} p^s_j x_{j,q} \leq \tilde{C}^s_{k'}, \qquad \forall k, k' \in \{1,...,n\}, k \geq k', \forall s \in \mathcal{S} \tag{6.16}$$

$$\sum_{j=0, j\neq i}^{n} z_{i,j,r} = y_{i,r}, \qquad \forall i \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{6.17}$$

$$\sum_{i=0, i\neq j}^{n} z_{i,j,r} = y_{j,r}, \qquad \forall j \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{6.18}$$

$$\sum_{r=1}^{n} y_{j,r} = 1, \qquad \forall j \in \{1,...,n\} \tag{6.19}$$

$$D^s_{0,r} \geq \tilde{C}^s_k - M(2 - x_{j,k} - y_{j,r}), \qquad \forall j,k,r \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.20}$$

$$D^s_{0,r} \geq D^s_{n+1,r-1}, \qquad \forall r \in \{2,...,n\}, \forall s \in \mathcal{S} \tag{6.21}$$

$$D^s_{j,r} \geq D^s_{i,r} + t^s_{i,j} - M(1 - z_{i,j,r}), \qquad \forall i \in \{0,...,n\}, \forall j,r \in \{1,...,n\}, i \neq j, \forall s \in \mathcal{S} \tag{6.22}$$

$$L_{\max} \geq D^s_{j,r} - d^s_j, \qquad \forall j,r \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.23}$$

$$z_{i,j,r} \in \{0,1\} \qquad \forall i,j \in \{0,...,n\}, i \neq j, \forall r \in \{1,...,n\} \tag{6.24}$$

$$y_{i,r} \in \{0,1\} \qquad \forall i,j \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{6.25}$$

$$D^s_{i,r} \geq 0 \qquad \forall i,j \in \{0,...,n\}, i \neq j, \forall r \in \{1,...,n\} \tag{6.26}$$

$$\tilde{C}^s_k \geq 0 \qquad \forall k \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{6.27}$$

et les contraintes (6.1), (6.2), (6.4). (6.28)

Les contraintes (6.17) et (6.18) sont des contraintes de conservation du flot qui stipulent que si un site i est visité par une tournée $\sigma(B_r)$ alors il existe un unique arc entrant et un unique

arc sortant sélectionné pour i dans la tournée $\sigma(B_r)$. Les contraintes (6.19) font en sorte qu'un site est visité par une seule tournée. Les contraintes (6.21)-(6.23) sont des contraintes liées aux scénarios. Pour tout scénario, les contraintes (6.19) et (6.21) font en sorte qu'une tournée $\sigma(B_r)$ commence après la fin d'exécution de toutes les tâches qu'elle transporte et après la fin de la tournée qui la précède $\sigma(B_{r-1})$, respectivement. Les contraintes (6.22) éliminent automatiquement les sous-tours et calculent les dates de livraison des tâches sur chaque scénario. Ainsi, la date de livraison d'une tâche J_j dans B_r sur un scénario s est égale à la date de livraison de la tâche J_i qui la précède plus le temps de transport entre i et j si l'arc (i, j) est sélectionné dans $\sigma(B_r)$. L'expression du retard maximum est donnée par les contraintes (6.23).
Ce modèle contient $O(n^3)$ variables binaires, $O(n^2|\mathcal{S}|)$ variables continues et $O(n^3|\mathcal{S}|)$ contraintes.

Une formulation alternative (noté *SRM2*) également basée sur le modèle d'ordonnancement robuste *SM1* est présentée. Dans cette formulation, les contraintes de la partie tournées de véhicules sont exprimées à travers des variables continues qui retournent de manière explicite les durées des tournées. Ainsi, les variables $D^s_{i,r}$ du modèle précédent *SRM* sont remplacées par les variables suivantes.

— $A^s_j \geq 0$ est le temps de transport du dépôt jusqu'au site j sachant la tournée qui visite j.
— u^s_r la date de départ de la tournée $\sigma(B_r)$ sur le scénario $s \in \mathcal{S}$

Minimiser $L_{\max}$

$$A^s_0 = 0, \qquad \forall s \in \mathcal{S} \tag{6.29}$$

$$A^s_j \geq A^s_i + t^s_{i,j} - M(1 - z_{i,j,r}), \qquad \forall i \in \{0, ..., n\}, \forall j, r \in \{1, ..., n\}, i \neq j, \forall s \in \mathcal{S} \tag{6.30}$$

$$u^s_r \geq \tilde{C}^s_k - M(2 - x_{j,k} - y_{j,r}), \qquad \forall j, k, r \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{6.31}$$

$$u^s_r \geq u^s_{r-1} + \sum_{i=0}^{n} \sum_{j=0}^{n} t^s_{i,j} z_{i,j,r-1}, \qquad \forall r \in \{2, ..., n\}, \forall s \in \mathcal{S} \tag{6.32}$$

$$L_{\max} \geq u^s_r + A^s_j - d^s_j - M(1 - y_{j,r}), \qquad \forall j, r \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{6.33}$$

$$u^s_r \geq 0 \qquad \forall r \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{6.34}$$

$$A^s_j \geq 0 \qquad \forall i \in \{0, ..., n\}, \forall s \in \mathcal{S} \tag{6.35}$$

et les contraintes (6.1), (6.2), (6.4), (6.27), (6.16), (6.17) - (6.19), (6.24), (6.25)

Les contraintes (6.29) fixent les temps de transport jusqu'au dépôt à 0 et les contraintes (6.30) donnent les durées de transport jusqu'aux différents sites. Les contraintes (6.31) et (6.32) assurent qu'une tournée commence après la fin de production du batch transporté et après la fin de la tournée qui la précède. Le retard maximum à minimiser est donné par les contraintes (6.33).
Ce modèle contient $O(n^3)$ variables binaires, $O(n|\mathcal{S}|)$ variables continues et $O(n^3|\mathcal{S}|)$ contraintes. A noter que le nombre de contraintes de ce modèle est inférieur a celui de *SRM* mais nécessite l'introduction de la constante grand M dans les contraintes (6.31).

Nous pouvons à présent combiner les deux formulations proposées pour la partie tournées de véhicules avec le modèle d'ordonnancement robuste *SM2* de sorte à obtenir deux formulations différentes pour le problème intégré d'ordonnancement et de tournées de véhicules robuste.

Soit *SRM3* le modèle qui combine les formulations les modèles *SM2* et *SRM1*. La nouvelle formulation est obtenue en retirant le $L_{\max}$ des contraintes de *SM2* en le remplaçant par les contraintes (6.17)–(6.26).

$$\text{Minimiser} \quad L_{\max}$$
$$\text{avec les contraintes } (6.1), (6.2), (6.4), (6.7), (6.8), (6.27), (6.17) - (6.26)$$

La quatrième formulation est une combinaison des modèles *SM2* pour la partie ordonnancement et *SRM2* pour la partie tournées de véhicules.

$$\text{Minimiser} \quad L_{\max}$$
$$\text{avec les contraintes (6.1), (6.2), (6.7), (6.8), (6.17) - (6.19), (6.32) - (6.35), (6.24), (6.25), (6.4)}$$

En utilisant le modèle *SM3* basé sur les contraintes de précédences, nous obtenons également deux nouvelles formulations pour le problème. La formulation qui suit notée *SM5* combine les modèles *SM3* et *SRM1*

$$\text{Minimiser} \quad L_{\max}$$
$$D^s_{0,r} \geq C^s_j - M(1 - y_{j,r}), \qquad \forall j \in \{1, ..., n\}, \forall r \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{6.36}$$
$$\text{avec les contraintes (6.9) - (6.12), (6.14), (6.15), (6.17) - (6.22), (6.23) - (6.26)}$$

Où les contraintes (6.36) donnent les dates de départ du véhicule. Les contraintes (6.36) étant équivalentes aux contraintes (6.20) qui comportent des variables positionnelles, le nombre de contraintes dans (6.36) est inférieur au nombre de contraintes (6.20) comportant des variables positionnelles.

Enfin, la dernière formulation résulte de la combinaison des modèles *SM3* et *SRM2* et est comme suit.

$$\text{Minimiser} \quad L_{\max}$$
$$u^s_r \geq C^s_j - M(1 - y_{j,r}), \qquad \forall j \in \{1, ..., n\}, \forall r \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{6.37}$$
$$\text{avec les contraintes (6.9) - (6.12), (6.32)-(6.30), (6.33)-(6.35), (6.17) - (6.19), (6.24), (6.25)}$$

où l'ensemble de contraintes (6.37) qui calculent les dates de départ du véhicule est équivalent à (6.31) et comporte également moins de variables pour les raisons citées précédemment.

Exemple Soit une instance du problème robuste d'ordonnancement et de tournées de véhicules avec deux scénarios. Les données du problème sont présentées dans les tableaux suivants.

$s=1$	J_1	J_2	J_3	J_4	J_5
r_j^1	0	7	3	4	3
p_j^1	3	4	1	2	4
d_j^1	11	18	9	10	17

$s=2$	J_1	J_2	J_3	J_4	J_5
r_j^2	3	3	0	1	7
p_j^2	2	5	1	3	3
d_j^2	9	17	10	11	18

Les matrices des durées de transport $(t_{i,j}^1)$ et $(t_{i,j}^2)$ sont comme suit :

$$(t_{i,j}^1) = \begin{pmatrix} 0 & 2 & 4 & 3 & 2 & 2 \\ 2 & 0 & 3 & 2 & 1 & 3 \\ 3 & 3 & 0 & 3 & 2 & 2 \\ 3 & 2 & 3 & 0 & 1 & 3 \\ 2 & 1 & 2 & 1 & 0 & 2 \\ 2 & 3 & 2 & 3 & 2 & 0 \end{pmatrix} \quad (t_{i,j}^2) = \begin{pmatrix} 0 & 3 & 2 & 2 & 2 & 3 \\ 3 & 0 & 4 & 1 & 2 & 3 \\ 2 & 4 & 0 & 4 & 3 & 2 \\ 2 & 1 & 4 & 0 & 2 & 4 \\ 2 & 2 & 3 & 2 & 0 & 4 \\ 3 & 3 & 2 & 4 & 4 & 0 \end{pmatrix}$$

Soit une solution du problème composée d'une séquence complète de production $\pi = (J_3, J_4, J_1, J_5, J_2)$, un ensemble de deux batchs $|B| = 2$ tels que $B_1 = \{J_1, J_3, J_4\}$ et $B_2 = \{J_2, J_5\}$ ainsi que des séquences de livraison $\sigma(B_1)$ et $\sigma(B_2)$ pour chacun des batchs telles que $\sigma(B_1) = (J_4, J_3, J_1)$ et $\sigma(B_2) = (J_2, J_5)$, i.e., les tournées pour les deux batchs sont respectivement $(J_4 \rightarrow J_3 \rightarrow J_1)$ et $(J_5 \rightarrow J_2)$. La figure 6.2 illustre les phases de production et de livraison sur les deux scénarios. La partie ordonnancement est représentée dans la figure par un diagramme de Gantt et la partie tournées de véhicules est représentée par des flèches dont la longueur indique la durée de transport. Les sites sont affichés entre parenthèses où 0 indique le dépôt et (j) tel que $j > 0$ indique le site de la tâche J_j. Le départ d'une nouvelle tournée est indiquée par une double barre $(||)$ et l'arrivée à un site est indique par une simple barre $(|)$. Ainsi, dans la solution illustrée dans la figure 6.2, le retard maximum $L_{\max}$ pour les deux scénarios est égale à 3. Le retard sur les scénarios est donné par la tâche J_2.

6.3 Algorithme tabou pour le problème intégré d'ordonnancement et tournées de véhicule robuste "standard"

Bien que nous ayons proposé des modèles linéaires en nombres entiers pour le problème robuste d'ordonnancement et de tournées de véhicules, ceux ci ne peuvent être utilisés que pour résoudre des instances de petites taille. Afin de pallier cette difficulté, nous proposons dans cette section deux algorithmes de recherche tabou. Le premier algorithme noté TS (Tabu search for robust Scheduling) est utilisé pour résoudre le problème d'ordonnancement robuste. Le second algorithme proposé noté TSR (Tabu search for robust Scheduling & Routing) est une extension de l'algorithme TS dans lequel la partie tournées de véhicules est introduite.

Dans les deux cas, une version standard de l'algorithme de recherche tabou est implémentée. Cette méta-heuristique a été introduite par [Glover, 1989] et fonctionne de la

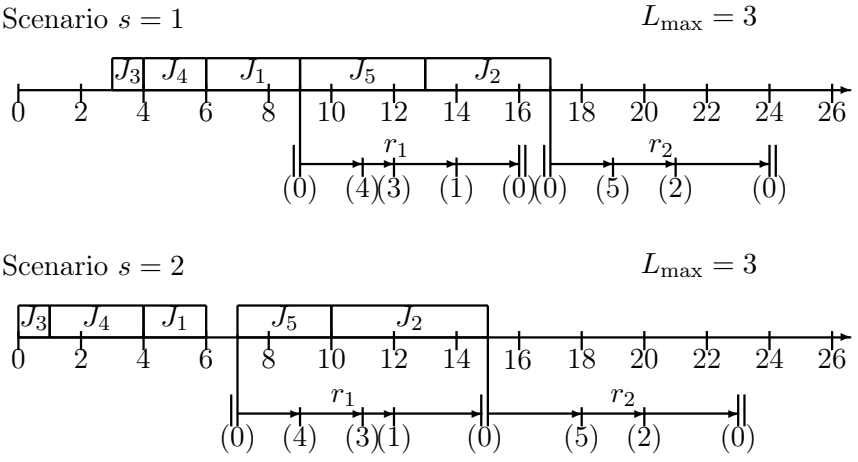


FIGURE 6.2 – Solution d'ordonnancement et tournées de véhicules robuste sur deux scénarios

manière suivante. Partant d'une solution initiale et en définissant une structure de voisinage, la procédure sélectionne la solution la plus appropriée parmi les solutions voisines. Une solution peut être sélectionnée si elle n'est pas dans la liste tabou ou ne satisfait pas un critère appelé "critère d'aspiration", autrement la solution est rejetée et la procédure cherche une nouvelle solution parmi les solutions voisines. La solution choisie est alors introduite dans la liste tabou et le processus est répété en utilisant la solution sélectionnée comme solution de base. Le processus s'arrête lorsque le critère d'arrêt est atteint. Nous commençons par présenter l'algorithme de recherche tabou pour le problème d'ordonnancement robuste ensuite nous présenterons une extension de l'algorithme TS prenant en compte la partie tournée de véhicules du problème intégré de production et de distribution.

Les heuristiques TS et TSR sont esquissées dans les sections 6.3 et 6.3 respectivement. Nous commençons par présenter une ébauche de l'algorithme TS avant de l'étendre au problème intégré d'ordonnancement et de tournées de véhicule robuste.

Algorithm 2 Algorithme de recherche tabou pour l'ordonnancement robuste (TS)

Calcul de la solution initiale
- — Générer une séquence de production en utilisant l'algorithme glouton (détails dans la section 6.3.1).

Tant que le critère d'arrêt n'est pas atteint
- — Partant de séquence de production actuelle, trouver la meilleure solution (séquence de production) non tabou parmi les solutions voisines (détails dans 6.3.3)
- — Stocker la solution retenue dans la liste tabou

Une solution de l'algorithme TSR se compose d'une séquence de production des tâches pour la partie ordonnancement, une partition des tâches en batchs et une séquence de livraison pour les tâches de chaque batch. Ainsi, l'heuristique se décompose en trois parties distinctes et constitue une extension de l'heuristique TS qui représente la première partie de l'algorithme.

Algorithm 3 Algorithme de recherche tabou pour l'ordonnancement et la distribution robuste (TSR)

Calcul de la solution initiale
- Générer une solution réalisable qui comprend une séquence de production, un regroupement des tâches en batchs et une séquence de livraison pour chaque batch. La solution initiale est calculée par l'algorithme glouton (détails dans 6.3.1).

Tant que le critère d'arrêt n'est pas atteint
- PARTIE 1. Trouver une séquence de production
  - La constitution des batchs ainsi que la séquence de livraison pour chacun d'eux étant fixées, trouver à partir de la séquence de production actuelle la meilleure solution (séquence de production) non tabou parmi les solutions voisines (détails dans 6.3.3).
  - Stocker la solution retenue dans la partie ordonnancement de la liste tabou.
- PARTIE 2. Trouver une partition des tâches en batchs
  - Partant de la constitution des batchs actuelle et de la séquence de production obtenue durant la première partie, trouver la meilleure partition des tâches parmi les solutions voisines. Afin d'évaluer une solution, une heuristique de plus proche voisin sur le scénario de référence s_0 est appliquée sur les batchs (détails dans 6.3.3).
  - Stocker la solution retenue dans la partie batching de la liste tabou.
- PARTIE 2. Trouver une séquence de livraison pour chaque batch
  - Partant de la séquence de production de la partie 1 et de la partition des tâches en batchs de la partie 2, trouver la meilleure permutation dans la livraison de chaque batch en utilisant l'heuristique 2-opt (simple méthode de descente, détails dans la section 6.3.3).

6.3.1 Solutions initiales

Afin d'initialiser les méthodes TS et TSR, deux heuristiques constructives sont utilisées. La solution construite pour initialiser l'heuristique TS notée GS (Greedy, Scheduling) est un simple tri des tâches selon la moyenne des dates de fin de production souhaitées sur le scénario de référence $s = 1$. Ainsi, la séquence de production initiale pour TS est obtenue en une seule étape en appliquant la règle EDD sur le scénario de référence $s = 1$. L'heuristique TSR nécessite quant à elle une solution initiale (noté GSR) composée d'une séquence de production des tâches, une partition des tâches en batchs et un ordre de livraison pour chaque batch. Les heuristiques GS et GSR utilisent les données du scénario de référence $s = 1$ afin de retourner les solutions initiales. L'heuristique GS étant inclue dans GSR, nous présentons dans de qui suit l'algorithme glouton GSR.

Algorithm 4 Algorithme glouton pour l'ordonnancement et la distribution robuste (GSR)

- Trier les tâches par ordre croissant sur les dates de fins souhaitées du scénario $s = 1$, i.e., $d^1_{\sigma_1} \leq d^1_{\sigma_2} \leq \cdots \leq d^1_{\sigma_n}$.
- Insérer la tâche J_{σ_1} dans le premier batch et de J_{σ_2} jusqu'à J_{σ_n}, la meilleure des deux options suivantes est sélectionnée pour chaque tâche :
  - Insérer la tâche dans le batch courant.
  - Créer un nouveau batch et insérer la tâche.

6.3.2 Codage de la solution

L'algorithme de recherche tabou sur le problème d'ordonnancement robuste TS étant inclu dans le l'algorithme intégré d'ordonnancement et de tournées de véhicule TSR, l'encodage de TSR comprend celui de TS. Etant donné que TSR comprend trois voisinages distincts qui correspondent aux trois parties de l'algorithme de recherche tabou (séquence de production, partition des tâches en batchs et séquence de livraison), le codage de chaque partie est traité indépendamment.

Encodage de la partie production : une solution de la partie production est encodée par un vecteur v de taille n où la case v_i correspond à la position de la tâche J_i dans la séquence de production.

Encodage de la partie batching : la partition des tâches en batchs est également représentée par un vecteur de taille n noté v' dans lequel une case v'_i contient le numéro de la tournée qui livre la tâche J_i.

Encodage de la partie tournées de véhicule : de la même manière, un troisième vecteur v'' de taille n est utilisé pour le codage de la séquence de livraison. Ainsi, la case v''_i contient le numéro de la tâche livrée en position i.

Un exemple de codage d'une solution de TSR est présenté dans la figure 6.3. Nous reprenons pour cette illustration la solution sur le scénario $s = 1$ de l'exemple 6.2.

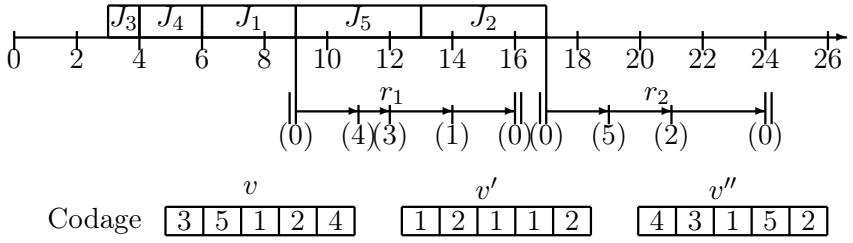


FIGURE 6.3 – Codage d'une solution du problème d'ordonnancement et de distribution robuste

6.3.3 Définition des voisinages

Nous présentons dans cette section les voisinages utilisés pour les deux algorithmes TS et TSR. D'après l'algorithme 6.3, le voisinage utilisé pour TS se compose d'une seule partie et consiste à trouver la meilleure séquence de production sur un voisinage donné. Ce voisinage se retrouve également dans l'algorithme 6.3 et constitue la première partie de l'algorithme. Ainsi, nous présentons dans ce qui suit le voisinage utilisé pour la partie production avant de passer au voisinage utilisé pour la partition des tâches en batchs. Enfin, nous passerons à la permutation utilisée pour la procédure 2-opt utilisée dans TSR. Un périmètre de recherche noté $\delta \in \{0, 1\}$ est utilisé dans les différents voisinages.

Voisinage pour la partie production : Deux types de voisinages sont utilisés.

- Permutation de deux positions dans la séquence : ce voisinage effectue l'échange de deux valeurs dans le vecteur v du codage, autrement dit la permutation de la position de deux tâches. Soient v_i et v_j les positions des tâches J_i et J_j dans la séquence de production, la permutation des valeurs de v_i et v_j n'est possible que si la position de la tâche J_j dans la séquence est dans le périmètre de la position de la tâche J_i, i.e., $|v_i - v_j| \leq \delta n$.
- Insertion d'une tâche à une position dans la séquence : étant donnée une tâche J_j, ce voisinage change sa position v_j en l'insérant à une position k telle que $|v_j - v_k| \leq \delta n$. Une telle insertion avance la position des tâches produites entre les positions v_j et v_k si $v_i > v_k$ et recule leurs positions si $v_i < v_k$.

Voisinage pour la partition des tâches en batchs : quatre voisinages sont utilisés.

- Permutation de batchs : ce voisinage effectue une permutation de deux valeurs dans le vecteur v' du codage. Soient v'_i et v'_j les numéros des batchs de J_i et J_j respectivement, la procédure cherche la meilleure solution voisine possible telle que $v'_i \neq v'_j$ et $v'_i - \delta|B| \leq v'_j \leq v'_i + \delta|B|$.
- Insertion d'une tâche dans un batch : ce voisinage permet d'insérer une tâche J_j du batch v'_i dans un batch existant v_k tel que $v'_i - \delta|B| \leq v_k \leq v'_i + \delta|B|$.
- Scission d'un batch : ce voisinage permet de scinder un batch en deux batchs successifs. Etant donné que le nombre de possibilités pour partitionner un batch en deux est exponentiel, les tâches d'un batch sont triées selon la rêgle EDD. Ainsi, le

nombre de possibilités se trouve réduit à la taille du batch.
— Fusion de deux batchs : ce voisinage effectue la fusion de deux batchs consécutifs. Soient B_i et B_{i+1} les deux batchs fusionnés, les batchs B_k tels que $k > i$ sont avancés d'une position.

Voisinage pour le problème de tournées de véhicule : pour ce dernier voisinage, nous avons utilisé l'opérateur 2-opt standard. Soient v''_i et v''_j les tâches produites en position i et j respectivement, ce mouvement effectue une permutation de deux valeurs du vecteur v'' dans le cas où les tâches σ_i et σ_j sont dans la même tournée, i.e., $v''_{\sigma_i} = v''_{\sigma_j}$.

6.3.3.1 Liste tabou

Afin d'éviter que l'algorithme tabou sélectionne des solutions récemment visitées, la liste tabou contient les solutions ne pouvant être sélectionnées, appelées solutions tabou. Dans notre cas, du fait que la partie ordonnancement et la partie batching sont traitées séparément, nous avons opté pour deux liste tabou de tailles n correspondant respectivement à la partie ordonnancement et à la partie batching. La première liste tabou contient des vecteurs v correspondant aux vecteurs v de la partie ordonnancement des dernières solutions visitées. De même pour la liste tabou de la partie batching, celle-ci contient les vecteurs v' des dernières solutions visitées. La partie tournées de véhicule ne nécessite pas de liste tabou du fait que l'algorithme 2-opt est appliqué comme une méthode de descente. Ainsi, une solution est considérée tabou dans deux cas. En effet, elle est tabou si le vecteur v de sa partie production se trouve dans la liste tabou ou le vecteur v' de sa partie batching se trouve dans la liste tabou.

6.3.3.2 Critère d'aspiration

Du fait qu'une solution est considérée tabou si sa partie production est tabou et si sa partie batching est tabou, nous introduisons un critère d'aspiration qui consiste à accepter des solutions tabou si la valeur de l'objectif pour ces solutions est meilleure que la solution optimale actuelle.

6.3.3.3 Conditions d'arrêt

L'heuristique tabou étant constituée de trois parties, des conditions d'arrêt sont utilisées pour chacune d'elles en plus de la condition d'arrêt globale. Ainsi, les parties ordonnancement, batching et tournées de véhicule s'arrêtent après un nombre d'itérations de la procédure sans améliorer la solution. La condition d'arrêt pour l'algorithme global est définie par l'atteinte d'un temps limite d'exécution.

6.4 Conclusion

Dans ce chapitre, nous avons proposé plusieurs modèles de programmation linéaire en nombres entiers pour résoudre un problème intégré d'ordonnancement et de tournées

robuste, selon les modèles par scénarios de Kouvelis et Yu [Kouvelis et Yu, 1997], correspondant a une vision "standard" de l'optimisation robuste. Pour traiter des instances de taille réaliste, nous proposons par ailleurs une méthode tabou, basée sur des voisinages simples. Ces méthodes seront comparées dans le chapitre 8 à une approche d'optimisation robuste avec récupération on-line, présentée dans le chapitre suivant.

Chapitre 7

Résolution du cas robuste par l'approche de récupération on-line

Ce chapitre exploite la structure des "groupes de tâches permutables" pour représenter une solution du problème d'ordonnancement et de distribution. Traditionnellement, les solutions de problèmes d'ordonnancement sont représentées par un ordre d'exécution des tâches sur chacune des machines. La structure de groupes de tâches permutables assigne un ordre sur un ensemble de tâches qui constituent des groupes, c'est-à-dire un ordre de groupe. Suivant les contraintes du problème, la permutation des tâches à l'intérieur d'un groupe doit rester réalisable. Une telle structure fournit plus de flexibilité à l'utilisateur final et permet particulièrement une meilleure réaction en s'adaptant aux événements inattendus. Selon le schéma d'optimisation robuste avec récupération, nous commençons par traiter la première étape qui consiste à trouver une séquence complète de production π, l'ensemble des batchs $\mathcal{B}$ et pour chacun des batchs $B \in \mathcal{B}$, une séquence de livraison complète. La séquence de production, la constitution des batchs et la séquence de livraison retournée par la première étape étant maintenues durant la seconde étape, l'algorithme de récupération *online* $A(x, s)$ s'adapte au scénario révélé. Ainsi, suivant la séquence prescrite et le scénario révélé, l'algorithme *online* détermine les dates de fin de production et de livraison des tâches.

7.1 Introduction

Nous présentons dans cette section une version plus flexible du modèle de récupération on-line (*online recoverable robustness model*) pour le problème d'ordonnancement ainsi que le problème intégré d'ordonnancement et de tournées de véhicules. De la même manière que le modèle précédent, nous supposons que la solution de la première étape $x = (\pi, \mathcal{B}, \Sigma)$ est constituée par une séquence de production π, une partition des tâches en batchs $\mathcal{B}$ et des séquences de livraison des batchs $\sigma(B)$. Cependant, nous considérons à présent que les séquences de production ainsi que les séquences de livraison de la solution de la première étape $x = (\pi, \mathcal{B}, \Sigma)$ sont partielles alors que la constitution des batchs $\mathcal{B}$ reste complète. En outre, nous considérons que les séquences partielles de production (et de livraison) sont exprimées sous la forme de groupes de tâches (sites) permutables, respectivement. En

d'autres termes, une séquence partielle impliquant un ensemble de E tâches est de la forme $g_1, g_2, \ldots, g_q$ où $\cup_{h=1}^{q} g_h = E$ et $\cap_{h=1}^{q} g_h = \emptyset$ et exprime également la relation de précédence $J_i \prec J_j$, $\forall i \in g_x$, $\forall j \in g_y$, $\forall x < y$. Par conséquent, π est une séquence de groupes de tâches permutables sur $\mathcal{J}$ tandis que $\sigma(B)$ est une séquence de tâches permutables sur $B \in \mathcal{B}$. Ainsi, l'ordre des tâches à l'intérieur de chaque groupe n'est pas déterminé durant la première étape. Dés lors, il apparaît que l'algorithme $A(x, s)$ doit déterminer la séquence complète pour chaque groupe afin de retourner les dates de fin de production des tâches et leurs dates de livraison. Dans la suite du chapitre, nous donnerons la définition de $A(x, s)$ puis nous présentons dans la section 7.2 un modèle PLNE de récupération on-line pour le problème d'ordonnancement robuste. L'approche proposée sera étendue au problème intégré d'ordonnancement et de tournées de véhicules dans la section 7.3.

7.2 Formulation PLNE du problème d'ordonnancement robuste avec récupération on-line

Dans cette section, nous considérons uniquement la partie ordonnancement du problème. Comme mentionné précédemment, l'algorithme de récupération est représenté par un algorithme glouton on-line. Etant donné une date t, l'algorithme de récupération on-line $A(x, s)$ a seulement connaissance des tâches disponibles à l'instant t sur un scénario s. Ainsi, l'information disponible pour $A(x, s)$ à l'instant t se restreint aux dates de début au plus tôt des tâches telles que $r_i^s \leq t$. En d'autres termes, soit J_j une tâche d'un groupe G_k dont la production est terminée, l'algorithme on-line sélectionne une des tâches disponibles du groupe G_k. Dans le cas où la production d'un groupe n'a pas commencé, la première tâche d'un groupe est sélectionnée de la même manière, i.e., une des tâches disponibles. Pour récapituler, à partir de la séquence de groupes retournée par la première étape, $A(x, s)$ ordonne les tâches à l'intérieur de chaque groupe suivant les dates de début au plus tôt en utilisant la règle (ERD) sur le scénario réalisé.

Pour simuler cette règle, une liste de successeurs et de prédécesseurs par rapport aux dates de début au plus tôt est préalablement calculée pour chaque tâche et chaque scénario. Soient $Succ_j^s$ et $Prec_j^s$ respectivement la liste de successeurs et de prédécesseurs d'une tâche J_j sur un scénario s définies comme suit.

$$Prec_j^s = \{J_i \in \mathcal{J} / (r_i^s < r_j^s) \text{ ou } (r_i^s = r_j^s \text{ et } i < j)\}$$

$$Succ_j^s = \{J_i \in \mathcal{J} / (r_i^s > r_j^s) \text{ ou } (r_i^s = r_j^s \text{ et } i > j)\}$$

Ces listes sont utilisées afin de s'assurer que la règle ERD est respectée dans chaque groupe et pour chaque scénario. Afin de modéliser la séquence de groupes, nous proposons un modèle PLNE. Les variables du modèle sont définies comme suit.

— $\chi_{j,k} \in \{0, 1\}$ vaut 1 si la tâche J_j est dans le groupe G_k, i.e., le groupe en position k, 0 sinon.

A noter que le nombre de groupes est borné par n (égal à n dans le cas où chaque groupe contient exactement une seule tâche). Nous pouvons présenter le PLNE noté OSM pour le problème d'ordonnancement robuste avec récupération on-line.

Minimiser $L_{\max}$

$$\sum_{k=1}^{n} \chi_{j,k} = 1, \qquad \forall j \in \{1,...,n\} \tag{7.1}$$

$$r_i^s + p_i^s + \sum_{l \in (Succ_i^s \cap Prec_j^s)} p_l^s \chi_{l,k} + p_j^s - M(2 - \chi_{i,k} - \chi_{j,k}) \leq C_j^s, \qquad \forall i,j,k \in \{1,...,n\}, j \in Succ_i^s, \forall s \in \mathcal{S} \tag{7.2}$$

$$r_i^s + p_i^s + \sum_{l \in Succ_i^s} p_l^s \chi_{l,k} + \sum_{q=k+1}^{k'-1} \sum_{l=1}^{n} p_l^s \chi_{l,q} + \sum_{l \in Prec_j^s} p_l^s \chi_{l,k'} + p_j^s - M(2 - \chi_{i,k} - \chi_{j,k'}) \leq C_j^s, \qquad \forall i,j,k,k' \in \{1,...,n\}, k' > k, \forall s \in \mathcal{S} \tag{7.3}$$

$$L_{\max} \geq C_j^s - d_j^s, \qquad \forall j \in \{1..n\}, \forall s \in \mathcal{S} \tag{7.4}$$

Les contraintes (7.1) affectent chaque tâche à exactement un groupe. Les contraintes (7.2) et (7.3) retournent la valeur minimum de la date de fin de production de chaque tâche pour chaque scénario suivant la définition de l'algorithme on-line. Etant donné une séquence de production, la date de fin de production au plus tôt d'une tâche J_j sur un scénario s est supérieure ou égale à la date de début au plus tôt r_i^s d'une tâche J_i ($J_i \in Prec_j^s$) plus la durée de production des tâches produites entre J_i et J_j dans la séquence (J_i et J_j inclues). Le cas où les tâches J_i et J_j sont dans le même groupe est traité par les contraintes (7.2) où l'expression $\sum_{l \in (Succ_i^s \cap Prec_j^s)} p_l^s \chi_{l,k}$ calcule la somme des durées de production des tâches entre J_i et J_j sur le scénario s suivant la règle ERD. Les contraintes (7.3) traitent le cas où J_i et J_j sont dans deux groupes différents où l'expression $\sum_{l \in Succ_i^s} p_l^s \chi_{l,k} + \sum_{q=k+1}^{k'-1} \sum_{l=1}^{n} p_l^s \chi_{l,q} + \sum_{l \in Prec_j^s} p_l^s \chi_{l,k'}$ calcule la somme des durées de production entre J_i et J_j. Les contraintes (7.4) calculent une borne supérieure du retard maximum dans le pire cas $L_{\max}$.
Ce modèle contient $O(n^2)$ variables binaires, $n|\mathcal{S}|$ variables continues et $O(n^4|\mathcal{S}|)$ contraintes.

Exemple

Nous reprenons l'exemple présenté dans la section 6.1. La solution optimale sur l'instance considérée comprend deux groupes $G_1 = \{J_1, J_3, J_4\}$ et $G_2 = \{J_2, J_5\}$ et un retard maximum dans le pire cas $L_{\max} = 0$. A noter que pour l'instance considérée, la version robuste présentée dans la section 6.1 a une solution optimale $L_{\max} = 5$. La figure 7.1 illustre la séquence obtenue sur les deux scénarios dans laquelle les tâches d'un même groupe sont ordonnées suivant la règle ERD.

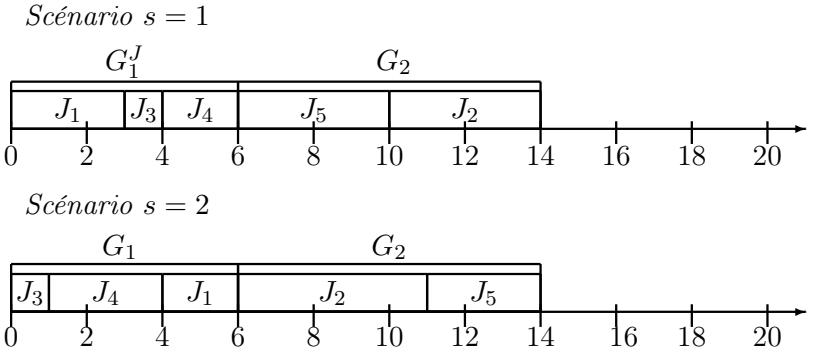


FIGURE 7.1 – Groupes $G_1^J = \{J_1, J_3, J_4\}$, $G_2^J = \{J_2, J_5\}$ et séquences obtenues sur les deux scénarios

7.3 Formulation PLNE du problème intégré d'ordonnancement et de tournées de véhicule robuste avec récupération on-line

Nous commençons par rappeler que la solution x de la première étape du problème intégré et robuste d'ordonnancement et tournées de véhicules où $x = (\pi, \mathcal{B}, \Sigma)$ est composée de :

- π : une séquence de groupes de tâches permutables
- $\mathcal{B}$: l'ensemble des batchs
- $\sigma(B)$: une séquence de livraison de groupes de tâches permutables sur un batch $B \in \mathcal{B}$.

De même, la définition de l'algorithme de récupération en ligne $A(x, s)$ sur la partie ordonnancement reste équivalente à celle présentée dans la section précédente sur le problème d'ordonnancement robuste. En d'autres termes, étant donné une séquence de production partielle π, l'algorithme $A(x, s)$ retourne pour chaque scénario une séquence complète de production en appliquant la règle ERD. En revanche, sur la partie livraison, l'algorithme $A(x, s)$ retourne une séquence de livraison pour chaque groupe de tâches permutables de livraison. Compte tenu des séquences, l'algorithme $A(x, s)$ retourne également les dates de production et de livraison au plus tôt. A noter que les groupes de production et de livraison sont indépendants et qu'un batch peut être constitué de plusieurs groupes de livraison. Etant donné une séquence partielle de livraison d'un batch qui consiste en une séquence de livraison de groupes d'un batch, nous considérons que l'algorithme de récupération on-line $A(x, s)$ retourne sur le scénario s la séquence de livraison en appliquant la règle du plus proche voisin. Le véhicule se trouvant à un site donné, l'algorithme cherche le plus proche voisin sur le scénario réalisé parmi tous les sites non visités du groupe concerné. Lorsque la livraison d'un groupe est terminée, l'algorithme cherche le plus proche voisin sur le scénario réalisé parmi les sites du groupe suivant (cette information est supposée disponible). Cette hypothèse possède un certain réalisme. En effet, les temps de parcours sont supposés incertains sur l'horizon de temps considéré mais lorsqu'on doit repartir immédiatement d'un site pour aller au suivant, on suppose que l'on connaît à ce moment là avec une précision suffisante le temps de parcours actuel du site où l'on se trouve vers tous les autres sites.

La formulation présentée dans cette partie est notée OSRM. La partie ordonnancement est basée sur le modèle d'ordonnancement robuste OSM. Les variables associées à la partie tournée de véhicules du problème sont comme suit.

- $\lambda^s_{j,k} \in \{0,1\}$ vaut 1 si la tâche J_j est livrée en position k sur le scénario s (i.e., J_j est la k^e tâche livrée sur s), 0 sinon.
- $\delta_{k,r} \in \{0,1\}$ vaut 1 si la tournée $\sigma(B_r)$ visite le site en position k (si la tâche J_j est dans le batch B_r), 0 sinon.
- $\gamma_{k,h} \in \{0,1\}$ vaut 1 si le site en position k est dans le groupe de livraisons permutables G^D_h, 0 sinon.
- $\theta_{h,r} \in \{0,1\}$ vaut 1 si le groupe de livraisons permutables G^D_h est livré par la tournée $\sigma(B_r)$ (i.e., $G^D_h \in B_r$), 0 sinon.
- $\alpha^s_r > 0$ la date de départ de la tournée $\sigma(B_r)$ sur le scénario s.
- $D^s_{j,r} > 0$ retourne la date de livraison de la tâche J_j pour un scénario $s \in \mathcal{S}$.

A partir de ces variables de décision, nous détaillons le modèle ORSM.

Nous reprenons pour cette formulation l'objectif ainsi que les contraintes (7.1) - (7.3) utilisés dans la formulation OSM. Comme mentionné dans la section 7.2, les contraintes (7.1) - (7.3) expriment la séquence de production des groupes de tâches permutables, la séquence de production à l'intérieur de chaque groupe en utilisant la règle ERD et les dates de fin de production des tâches sur chaque scénario.

Les contraintes relatives à la partie tournées de véhicules sont comme suit.
Les contraintes (7.5) et (7.6) ci-dessous bornent les dates de départ des tournées. C'est ainsi que les contraintes (7.5) assurent que la date de départ du véhicule pour livrer le batch B_r est supérieure aux dates de fin de production des tâches du batch B_r. Les contraintes (7.6) assurent que le véhicule est disponible à la date de départ d'une tournée. Ainsi, pour un scénario s, la tournée qui livre un batch B_r commence après la date de retour de la tournée qui livre le batch B_{r-1} où, la date de retour du véhicule au dépôt est égale à la date de livraison de la dernière tâche J_j de la tournée B_{r-1} plus le temps de transport jusqu'au dépôt.

$$\alpha^s_r \geq C^s_j + M(2 - \delta_{kr} - \lambda^s_{jk}), \quad \forall j, r, k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{7.5}$$

$$\alpha^s_r \geq D^s_j + t^s_{j0} + M(2 - \delta_{k,r-1} - \lambda^s_{jk}), \quad \forall j, k \in \{1, ..., n\}, \forall r \in \{2, ..., n\}, \forall s \in \mathcal{S} \tag{7.6}$$

Les contraintes (7.7) à (7.12) qui suivent donnent les dates de livraison des tâches suivant les dates de départ du véhicule, la composition des batchs et des groupes de livraison, la séquence de groupes de livraison dans chaque batch ainsi que la séquence de livraison des tâches à l'intérieur de chaque groupe de livraison calculée par un algorithme de plus proche voisin. Les contraintes (7.7) font en sorte que sur un scénario s, une tâche J_j en position k soit livrée après la tâche J_i en position $k-1$.

$$D^s_j \geq D^s_i + t^s_{ij} + M(2 - \lambda^s_{i,k} - \lambda^s_{j,k+1}), \forall i, j \in \{1, ..., n\}, j \neq i, \forall k \in \{1, ..., n-1\}, \forall s \in \mathcal{S} \tag{7.7}$$

Les contraintes (7.8) garantissent que la date de livraison de la tâche J_j en position k sur un scénario s est supérieure à la date de début de la tournée qui livre B_r plus le temps de transport entre le dépôt et le site j si J_j est dans B_r.

$$D^s_j \geq \alpha^s_r + t^s_{0j} - M(2 - \delta_{kr} - \lambda^s_{j,k}), \quad \forall j, r, k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{7.8}$$

Les contraintes (7.9) assurent que la règle du plus proche voisin sur chaque scénario est respectée à l'intérieur de chaque groupe de livraison. Soient J_i et J_j deux tâches du groupe G_h^D livrées respectivement en position $k-1$ et k sur le scénario s, les contraintes (7.9) garantissent que le site J_j est le plus proche voisin de i parmi toutes les tâches de G_h^D pas encore livrées.

$$D_j^s \leq D_i^s + t_{il}^s + M(6 - \gamma_{k-1,h} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s),$$
$$\forall i,j,l,h \in \{1,...,n\}, j \neq i, l \neq i, \forall k \in \{2,...,n\}, \forall k' \in \{k,...,n\}, \forall s \in \mathcal{S} \quad (7.9)$$

Les contraintes (7.10) assurent que la première tâche visitée d'un groupe est la plus proche de la tâche qui la précède dans le cas où celle ci n'est pas la première visitée dans la tournée. Ainsi, ces contraintes nous assurent que deux tâches J_i et J_j des groupes G_{h-1}^D et G_h^D du batch B_r et livrées en position $k-1$ et k respectivement sont telles que J_j soit la plus proche de J_i parmi tous les sites du groupe G_h^D.

$$D_j^s \leq D_i^s + t_{il}^s + M(8 - \gamma_{k-1,h-1} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s - \theta_{h-1,r} - \theta_{h,r}),$$
$$\forall i,j,l,r \in \{1,...,n\}, j \neq i, l \neq i, \forall h,k \in \{2,...,n\}, \forall k' \in \{k,...,n\}, \forall s \in \mathcal{S} \quad (7.10)$$

Etant donnée une tâche J_j du groupe G_h^D incluse dans le batch B_r telle que J_j soit la première tâche livrée dans la tournée livrant B_r, les contraintes (7.11) garantissent que sur tout scénario s, J_j est la plus proche du dépôt parmi toutes les tâches de G_h^D en s'assurant que G_h^D est le premier groupe de $\sigma(B_r)$, où $r \geq 2$.

$$D_j^s \leq \alpha_r^s + t_{0l}^s + M(8 - \gamma_{k-1,h-1} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s - \theta_{h-1,r-1} - \theta_{h,r}),$$
$$\forall i,j,l \in \{1,...,n\}, j \neq i, l \neq i, \forall r,h,k \in \{2,...,n\}, \forall k' \in \{k,...,n\}, \forall s \in \mathcal{S} \quad (7.11)$$

Les contraintes (7.12) garantissent quant à elles que la première tâche livrée dans la première tournée soit la plus proche du dépôt parmi toutes les tâches de son groupe.

$$D_j^s \leq \alpha_1^s + t_{0l}^s + M(4 - \gamma_{1,1} - \lambda_{j,1}^s - \gamma_{k,1} - \lambda_{l,k}^s), \quad \forall j,l,k \in \{1,...,n\}, \forall s \in \mathcal{S} \quad (7.12)$$

Etant donné que les variables $\gamma_{k,h}$ dépendent des positions des tâches, les contraintes (7.13) nous assurent qu'un groupe G_h^D contient les mêmes tâches quel que soit le scénario.

$$\gamma_{k,h} \geq 1 - M(3 - \lambda_{j,k}^{s'} - \lambda_{j,k'}^{s''} - \gamma_{k',h}),\ \forall j,h,k,k' \in \{1,...,n\}, k \neq k', \forall s', s'' \in \mathcal{S}, s'' > s' \quad (7.13)$$

Les contraintes (7.14)-(7.22) garantissent la faisabilité des tournées. Pour un scénario $s \in \mathcal{S}$, chaque tâche est affectée à une seule position (7.14) et chaque position à une seule tâche (7.15). Les contraintes (7.16), (7.17) et (7.18) nous assurent respectivement qu'une position est affectée à un seul groupe, à une seule tournée et que chaque groupe de livraison est associé à une seule tournée. Les contraintes (7.19) font que les tournées sont consécutives en assurant que si la tâche en position k est livrée par la tournée en position r alors la tâche en position $k+1$ est livrée par r ou $r+1$. Suivant le même principe, les contraintes (7.20) font en sorte que les groupes de livraison soient consécutifs. Les contraintes (7.21) assurent la cohérence de l'affectation des tâches aux positions, des positions aux groupes de livraison et des groupes de livraison aux tournées. Ainsi, si une position k est affectée au groupe G_h^D et le groupe G_h^D est affecté à la tournée r alors la position k est affectée

à la tournée r. Les contraintes (7.22) affectent la première position au premier groupe de livraison et à la première tournée ainsi que le premier groupe de livraison à la première tournée. Enfin, les contraintes (7.23) bornent le $L_{\max}$.

$$\sum_{j=1}^{n} \lambda_{j,k}^{s} = 1, \forall k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{7.14}$$

$$\sum_{k=1}^{n} \lambda_{j,k}^{s} = 1, \forall j \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{7.15}$$

$$\sum_{h=1}^{n} \gamma_{k,h} = 1, \forall k \in \{1, ..., n\} \tag{7.16}$$

$$\sum_{r=1}^{n} \delta_{k,r} = 1, \forall k \in \{1, ..., n\} \tag{7.17}$$

$$\sum_{r=1}^{n} \theta_{h,r} = 1, \forall h \in \{1, ..., n\} \tag{7.18}$$

$$\delta_{kr} + \delta_{k+1,r'} \leq 1, \forall k \in \{1, ..., n-1\}, \forall r, r' \in \{1, ..., n\}, r' \neq r, r' \neq r+1 \tag{7.19}$$

$$\gamma_{kh} + \gamma_{k+1,h'} \leq 1, \forall k \in \{1, ..., n-1\}, \forall h, h' \in \{1, ..., n\}, h' \neq h, h' \neq h+1 \tag{7.20}$$

$$\delta_{kr} \geq \gamma_{kh} + \theta_{hr} - 1, \forall k \in \{1, ..., n-1\}, \forall h, h' \in \{1, ..., n\}, h' \neq h, h' \neq h+1 \tag{7.21}$$

$$\delta_{11} = 1, \gamma_{11} = 1, \theta_{11} = 1, \tag{7.22}$$

$$L_{\max} \geq D_j^s - d_j^s, \forall j \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{7.23}$$

Ce modèle contient $O(n^2|\mathcal{S}|)$ variables binaires, $n|\mathcal{S}|$ variables continues et $O(n^7|\mathcal{S}|)$ contraintes.

Exemple Nous considérons dans cet exemple l'instance présentée dans la section 6.2. La solution optimale sur cet exemple est comme suit.

- Deux groupes d'opérations permutables G_1^J et G_2^J sur la partie production tels que $G_1^J = \{J_1, J_3, J_4\}$ et $G_1^J = \{J_2, J_5\}$,
- deux batchs pour la partie tournées de véhicules $B_1 = \{J_1, J_3, J_4\}$ et $B_2 = \{J_2, J_5\}$
- le batch B_1 est composé de deux groupes $G_1^D = \{J_3\}$ et $G_2^D = \{J_1, J_4\}$ et le deuxième batch B_2 est composé d'un seul groupe de livraison $G_3^D = \{J_2, J_5\}$.

La phase de livraison est illustrée sur les deux scénarios dans la figure 7.3 et la solution globale du problème incluant le problème d'ordonnancement et le problème de tournée de véhicules est illustrée pour les deux scénarios dans la figure 7.3. Le retard maximum dans le pire des cas sur cet exemple est $L_{\max} = 0$. A noter que pour l'instance considérée, la version robuste présentée dans la section 6.2 retourne une solution optimale $L_{\max} = 3$.

7.4 Algorithme tabou pour le problème intégré d'ordonnancement et tournées de véhicule robuste avec récupération on-line

Les modèles linéaires en nombres entiers proposés pour le problème d'ordonnancement robuste et flexible (OSM) ainsi que celui proposé pour le problème intégré d'ordonnancement et de tournées de véhicule robuste et flexible (OSRM) ne peuvent être utilisés pour la résolution d'instances de tailles raisonnables. En effet, l'intégration de la flexibilité par

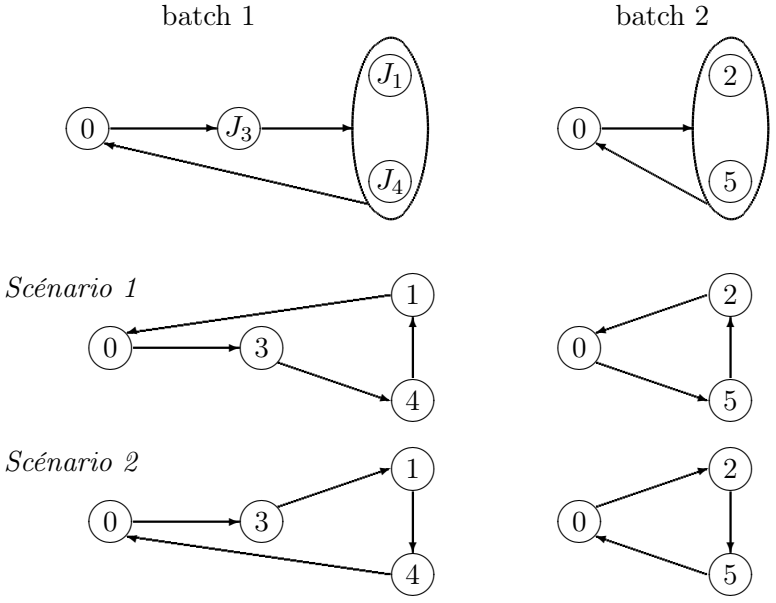


FIGURE 7.2 – Séquence de groupes de livraison $\{J_3\}, \{J_1, J_4\}$ et $\{J_2, J_5\}$ produisant des tournées différentes sur les deux scénarios

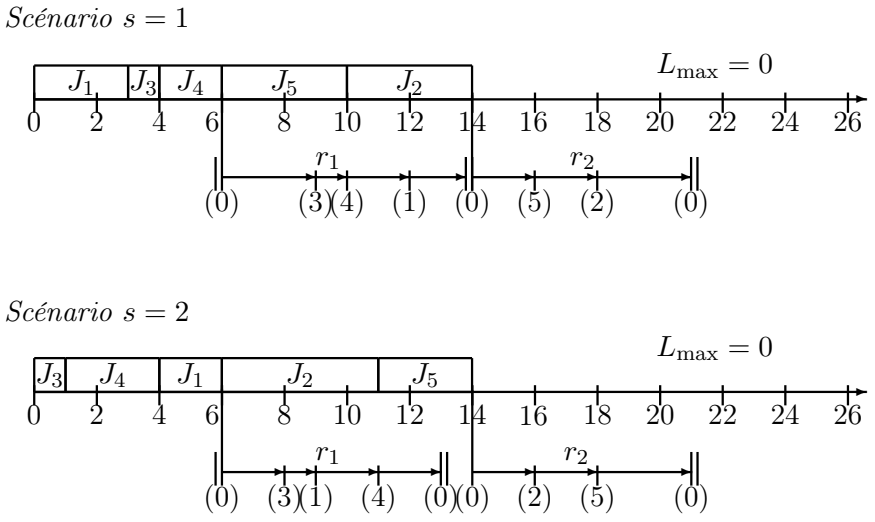


FIGURE 7.3 – Phases de production et de livraison sur les deux scénarios possibles

l'intermédiaire des groupes d'opérations permutables argumente de manière importante la taille des formulations. De même que pour la version robuste du problème d'ordonnancement et le problème intégrant la partie distribution, nous proposons dans cette section un premier algorithme de recherche tabou pour le problème d'ordonnancement robuste et flexible avec récupération noté TOS (*Tabu search for robust Online Scheduling*). Ensuite, nous étendons l'heuristique TOS au problème intégré d'ordonnancement et de tournées de véhicule robuste avec récupération noté TOSR. Etant donné que le modèle OSM correspondant au problème d'ordonnancement robuste avec récupération on-line utilise la notion de groupe de tâches permutables, l'heuristique TOS lui correspondant diffère de l'heuristique TS proposée pour le problème d'ordonnancement robuste "standard". De la même manière, les groupes d'opérations permutables étant utilisés pour la livraison, la partie tournées de véhicule de TOSR diffère de celle de TSR proposée pour le problème intégré d'ordonnancement et de distribution robuste "standard" (voir l'algorithme 6.3). Dans ce qui suit, nous commençons par esquisser l'heuristique TOS avant de passer à l'heuristique TOSR.

Algorithm 5 Algorithme de recherche tabou pour l'ordonnancement avec récupération on-line (TOS)

Calcul de la solution initiale
- Générer une séquence de groupes de production en utilisant l'algorithme glouton (détails dans 7.4.1).

Tant que le critère d'arrêt n'est pas atteint
- Partant de la séquence de groupes de production actuelle, trouver la meilleure partition des tâches en groupes ainsi qu'une séquence de groupes non tabou parmi les solutions voisines (détails dans 7.4.3)
- Stocker la solution retenue dans la liste tabou

Ainsi, une solution de TOS consiste à trouver une solution de groupes d'opérations permutables alors qu'une solution de TOSR se compose d'une séquence de production de groupes également, d'une partition des tâches en batchs indépendamment des groupes de production et de livraison et enfin d'une séquence de groupes de livraison pour chaque batch.

Comme nous pouvons le voir, les heuristiques TSR et TOSR diffèrent sur plusieurs points. En effet, la notion de groupes d'opérations permutables apporte de la flexibilité à la partie production et la partie distribution. Cependant, afin de pouvoir comparer les deux heuristiques et pour simplifier l'implémentation nous avons considéré dans TOSR qu'un batch constitue également un groupe de livraison. Ainsi, la séquence de livraison de chaque batch dans TOSR est calculée par un simple algorithme glouton de plus proche voisin s'adaptant au scénario (l'algorithme $A(x, s)$) alors que celle-ci est optimisée dans TSR grâce à l'algorithme 2-opt appliqué à chaque batch mais sans connaître à l'avance le scénario réalisé. De cette manière, l'algorithme TOSR a l'avantage de la flexibilité étant donné que celui-ci s'adapte au scénario réalisé et adapte l'heuristique du plus proche voisin sur chaque scénario. L'heuristique TSR quant à elle n'est pas flexible puisque la séquence de livraison ne dépend pas du scénario réalisé mais par contre celle-ci est optimisée grâce à l'algorithme 2-opt.

Algorithm 6 Algorithme de recherche tabou pour l'ordonnancement et la distribution robuste avec récupération on-line (TOSR)

Calcul de la solution initiale
- Générer une solution réalisable qui comprend une séquence de groupes de production, un regroupement des tâches en batchs indépendamment des groupes et une séquence de livraison pour chaque batch. La solution initiale est calculée par l'algorithme glouton (détails dans 7.4.1).

Tant que le critère d'arrêt n'est pas atteint
- PARTIE 1. Trouver une séquence de production
  - La constitution des batchs ainsi que la séquence de livraison pour chacun d'eux étant fixées, à partir de la séquence de production actuelle, trouver la meilleure partition des tâches en groupes ainsi qu'une séquence de groupes non tabou parmi les solutions voisines (détails dans 7.4.3).
  - Stocker la solution retenue dans la partie ordonnancement de la liste tabou.
- PARTIE 2. Trouver une partition des tâches en batchs
  - Partant de la constitution des batchs actuelle et de la séquence de production de groupes obtenue durant la première partie, trouver la meilleure partition des tâches parmi les solutions voisines. Afin d'évaluer une solution, l'heuristique de plus proche voisin sur le scénario réalisé est appliquée (détails dans 7.4.3).
  - Stocker la solution retenue dans la partie batching de la liste tabou.

7.4.1 Solutions initiales

Afin d'initialiser les méthodes TOS et TOSR, nous proposons dans ce qui suit deux heuristiques constructives. La première heuristique noté GOS (*Gready, On-line, Scheduling*) construite pour initialiser TOS est un simple tri des tâches par rapport aux dates de fin de production souhaitées sur le scénario de référence $s = 1$. Dans GOS, un groupe d'opérations permutables contient une seule tâche et ainsi, la séquence de production initiale pour TOS est obtenue en une seule étape en appliquant la règle EDD sur le scénario de référence $s = 1$. L'initialisation de l'heuristique TOSR consiste quant à elle à trouver une séquence de production de groupes ainsi qu'une partition des tâches en batchs. Etant donné que la séquence de production est obtenue par l'algorithme de plus proche voisin sur le scénario réalisé, la partie batching comprend implicitement la partie livraison.

7.4.2 Codage de la solution

Dans l'algorithme TOSR, la partie tournée de véhicules est systématiquement résolue par l'algorithme du plus proche voisin sur le scénario réalisé. De plus, étant donné que nous supposons qu'un batch contient un seul groupe de livraisons permutables, il suffit de considérer uniquement la partie ordonnancement et la composition des batchs pour coder une solution. Etant donné qu'une solution de TOS correspond à la partie ordonnancement de TOSR, nous présentons dans ce qui suit le codage de chaque partie (ordonnancement et partition des tâches en batchs).

Codage de la partie production : une solution de la partie production est codée par un vecteur v de taille n où la case v_i correspond au groupe de la tâche J_i (les groupes

Algorithm 7 Algorithme glouton pour l'ordonnancement et la distribution robuste avec récupération on-line (GOSR)

- Trier les tâches par ordre croissant selon la moyenne des dates de fin souhaitées sur tous les scénarios, i.e., $\sum_{s\in\mathcal{S}} d^s_{\sigma_1} \leq \sum_{s\in\mathcal{S}} d^s_{\sigma_2} \leq \cdots \leq \sum_{s\in\mathcal{S}} d^s_{\sigma_n}$.
- Insérer la tâche J_{σ_1} dans le premier groupe et le premier batch et de J_{σ_2} jusqu'à J_{σ_n}, la meilleure des quatre options suivantes est sélectionnée pour chaque tâche :
  - Insérer la tâche dans le groupe courant ; insérer la tâche dans le batch courant.
  - Insérer la tâche dans le groupe courant ; créer un nouveau batch et insérer la tâche.
  - Créer un nouveau groupe et insérer la tâche ; insérer la tâche dans le batch courant.
  - Créer un nouveau groupe et insérer la tâche ; créer un nouveau batch et insérer la tâche.

d'opérations permutables sont traités successivement, i.e., si $v_i < v_j$ alors J_i est produite avant J_j).

Codage de la partie batching : celui-ci est similaire à celui utilisé dans l'algorithme TSR et est représenté par un vecteur de taille n noté v' dans lequel une case v'_i contient le numéro de la tournée qui livre la tâche J_i.

Un exemple de codage d'une solution de TOSR est présenté dans la figure 7.4. Nous reprenons pour cette illustration la solution sur le scénario $s = 1$ de l'exemple 7.3. Etant donné qu'un batch constitue un groupe de livraisons permutables, la règle du plus proche voisin est appliquée sur toutes les tâches d'un batch.

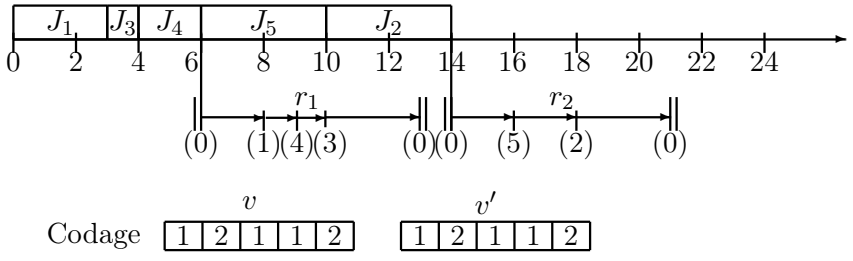


FIGURE 7.4 – Codage d'une solution du problème d'ordonnancement et de distribution robuste et flexible

7.4.3 Définition des voisinages

D'après l'algorithme 6.3 de TOSR, deux groupes de voisinages sont utilisés. Le premier consiste à trouver la composition des groupes de tâches permutables pour la production et la séquence de production des groupes et le second voisinage consiste à partitionner les tâches en batchs. Le second voisinage qui consiste à trouver une partition des tâches en batchs étant similaire à celui utilisée pour l'algorithme TSR, nous présentons uniquement

le voisinage de la partie production. A noter que le voisinage utilisé pour la partie production de TOSR est équivalent à celui de TOS et que périmètre de recherche $\delta \in \{0, 1\}$ (présenté dans la définition des voisinages de l'heuristique TSR dans la section 6.3.3) est également utilisé dans les différents voisinages.

Voisinage pour la partie production : quatre types voisinages sont utilisés pour partitionner les tâches en groupes.

- Permutation de groupes : ce voisinage effectue une permutation de deux valeurs dans le vecteurs v du codage. Soient v_i et v_j les numéros des groupes de J_i et J_j respectivement, la procédure cherche la meilleure solution voisine possible telle que $v_i \neq v_j$ et $v_i - \delta|G^J| \leq v_j \leq v_i + \delta|G^J|$.
- Insertion d'une tâche dans un groupe : ce voisinage permet d'insérer une tâche J_j du groupe v_i dans un groupe existant v_k tel que $v_i - \delta|G^J| \leq v_k \leq v_i + \delta|G^J|$.
- Scission de groupe : ce voisinage permet de scinder un groupe en deux groupes successifs. Etant donné que le nombre de possibilités pour partitionner un groupe en deux est exponentiel, les tâches d'un groupe sont triées selon l'ordre des dates de livraison souhaitées moyennes sur tous les scénarios. Ainsi, le nombre de possibilités se trouve réduit à la taille du groupe.
- Fusion de deux groupes : ce voisinage effectue la fusion de deux groupes consécutifs. Soient G_i^J et G_{i+1}^J les deux groupes fusionnés, les groupes G_k^J tels que $k > i$ sont avancés d'une position.

Les listes tabou, le critère d'aspiration ainsi que les conditions d'arrêt étant identiques à ceux utilisés pour la résolution de l'heuristique TSR, nous invitons le lecteur à se référer aux sections (6.3.3.1, 6.3.3.2 et 6.3.3.3) respectivement.

7.5 Conclusion

Nous abordons dans ce chapitre le problème intégré d'ordonnancement et de tournées de véhicules par une approche robuste avec récupération on-line. Pour ce problème, nous avons proposé de nouvelles formulations de programmation linéaire en nombres entiers. La difficulté de modélisation était d'intégrer dans la formulation le comportement de l'algorithme on-line qui s'adapte au scénario réalisé. L'algorithme on-line qui a été mis en oeuvre consiste à appliquer la règle ERD (première tâche disponible d'abord) pour la production et la règle du plus proche voisin pour la livraison. Nous proposons également une méthode tabou pour la résolution d'instances de tailles raisonnables. La comparaison avec la méthode tabou résolvant le problème d'optimisation robuste "standard" est présentée dans le chapitre suivant.

Chapitre 8

Expérimentations

Afin de valider nos modèles, les méthodes basées sur les séquences de groupes sont comparées à l'approche standard d'ordonnancement robuste basée sur une séquence totale de tâches. Nous présentons dans ce chapitre des résultats expérimentaux sur des instances générées aléatoirement. Nous commençons par présenter la génération des données avant de passer aux tests expérimentaux du problème d'ordonnancement robuste sans transport ainsi que les résultats obtenues sur le problème intégré d'ordonnancement et de tournées de véhicules robuste.

8.1 Génération des données

Les données du problème intégré de production et de distribution sous incertitudes sont générées de la manière suivante. Nous commençons par générer aléatoirement une instance du problème pour le scénario de référence $s = 1$. Pour ce faire, les durées d'exécutions p_j^1 sont générées aléatoirement dans intervalle $[1, 50]$. Soit P la somme des durées d'exécution sur le scénario de référence : $P = \sum_{j=1}^{n} p_j^1$. Les dates de début au plus tôt sont générées aléatoirement dans $[0, \gamma P]$ où γ est un paramètre donné. Etant donné des paramètres α et β, les dates de livraison souhaitées (dates de fin de production souhaitées dans le cas où la partie distribution n'est pas prise en compte) des tâches sont générées aléatoirement dans l'intervalle $[\alpha - \frac{\beta}{2}P, \alpha + \frac{\beta}{2}P]$. Concernant la partie distribution, une matrice des distances entre les différents sites est obtenue en générant aléatoirement des coordonnés (X_j^1, Y_j^1) dans l'intervalle $[1, 50]$ pour chaque site $j \in \{1, \ldots, n\}$. La distance entre deux sites i et j est alors obtenue en calculant la distance euclidienne entre i et j.

$$t_{i,j}^1 = t_{j,i}^1 = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$$

Afin de représenter l'incertitude sur les données, les instances sur les scénarios s tels que $s \geq 2$ sont générées de la façon suivante.

- r_j^s est généré dans $[(1 - \omega^r)r_j^1, (1 + \omega^r r_j^1)]$ où le coefficient ω^r est donné.
- p_j^s est généré dans $[(1 - \omega^p)p_j^1, (1 + \omega^p p_j^1)]$ où le coefficient ω^p est donné.
- d_j^s est généré dans $[(1 - \omega^d)d_j^1, (1 + \omega^d d_j^1)]$ où le coefficient ω^d est donné.
- X_j^s est généré dans $[(1 - \omega^X)X_j^1, (1 + \omega^X X_j^1)]$ où le coefficient ω^X est donné.

— Y_j^s est généré dans $[(1-\omega^Y)Y_j^1, (1+\omega^Y Y_j^1)]$ où le coefficient ω^Y est donné.

Le changement des coordonnées d'un site peut paraître surprenant mais les coordonnées ne doivent pas être prises comme des coordonnées géographiques réelles mais comme un moyen de générer une matrice de temps de transport cohérente. Pour des raisons pratiques, nous considérons que les coefficient $\omega^r, \omega^p, \omega^d, \omega^X$ et ω^Y sont identiques et notés w.

8.2 Résultats expérimentaux

Nous évaluons dans cette section les modèles linéaires en nombres entiers ainsi que les algorithmes tabou présentés dans les deux chapitres précédents. Nous commençons par présenter les résultats obtenus sur le problème d'ordonnancement avant de passer aux résultats du problème intégré d'ordonnancement et de distribution. Dans chacun des cas, les algorithmes du problème robuste "standard" sont comparés à ceux du problème robuste avec récupération on-line.

8.2.1 Résultats sur le problème ordonnancement

Nous évaluons dans cette partie les résultats obtenus pour le problème d'ordonnancement seul. Les tests sont effectués sur les instances présentées précédemment dans lesquelles les paramètres γ, α et β prennent les valeurs $\{0,5\}, \{1\}$ et $\{1\}$ respectivement. Le coefficient de variation ω prend sa valeur dans l'ensemble $\{0,2,\ 0,4,\ 0,6\}$. Pour chaque couple (n,s) 10 instances sont générées où $n \in \{10,25,40,100\}$ et $\mathcal{S} \in \{2,5,10\}$. Les tests sont exécutés sur un total de 360 instances sur une machine Intel i7-4770 CPU 3.40GHz avec 8GO de mémoire vive.

Nous évaluons et comparons les résultats obtenus par les modèles linéaires en nombres entiers ainsi que par les algorithmes tabou. Afin de valider les heuristiques proposées, celles-ci sont comparées aux résultats obtenus par les MILP correspondants. Ainsi, les résultats retournés par l'heuristique de recherche tabou TS sur le problème d'ordonnancement robuste sont comparés à ceux obtenues par le modèle SM proposé dans la section 6.1 pour le problème d'ordonnancement robuste "standard". De la même manière, les résultats de l'heuristique TOS sur le problème d'ordonnancement robuste avec récupération on-line sont comparés à ceux obtenues par le modèle OSM proposé dans la section 7.2 pour le problème d'ordonnancement robuste avec récupération on-line. Enfin, nous comparons les approche robustes "standard" avec la version intégrant la récupération on-line.

Soient $z_E(E)$ et $z_H(H)$ respectivement les valeurs exacte et heuristique de la fonction objectif retournées par l'algorithme $E \in \{SM, OSM\}$ et le PLNE de recherche tabou $H \in \{TS, TOS\}$.

Le tableau 8.1 fournit les résultats agrégés pour chaque valeur de n. Les statistiques prennent en compte la moyenne du temps CPU noté *ttb* pour atteindre la meilleure solution retournée par chacun des algorithmes. L'écart moyen entre la solution exacte (optimale) trouvée et la solution retournée par l'algorithme tabou est donné par les colonnes *gap*. Ainsi les colonnes *gap* retournent la valeur moyenne $(z_H(TS) - z_E(SM))/z_H(TS)$ dans le cas de l'approche robuste "standard" (noté ARS) et $(z_H(TOS) - z_E(OSM))/z_H(TOS)$ dans le cas de l'approche par groupe (noté ARG). Enfin, la dernière colonne Δ retourne

l'écart moyen $\Delta = (z_H(TS) - z_H(TOS))/z_H(TS)$ entre les deux heuristiques tabou.

	ARS		ARG		
n	$ttb(s)$	gap	$ttb(s)$	gap	Δ
10	0.01	1.7%	0.01	0%	24.79%
25	0.14	2,3%	0.15	-	17.28%
40	0.39	-	0.52	-	16.83%
100	3.21	-	12.58	-	13.71%

TABLE 8.1 – Comparaisons sur le problème d'ordonnancement sous incertitudes

Ce tableau montre les limites des modèles linéaires en nombres entiers et en particulier sur l'approche robuste avec récupération on-line. Sur les colonnes *ttb*, nous pouvons voir que le temps nécessaire à l'heuristique tabou TOS pour trouver la meilleure solution croit plus vite que celui nécessaire pour l'algorithme TS. Cependant, la différence entre les temps d'exécution des deux algorithmes n'est pas très significative et reste comparable sur les instances considérées. Enfin, la colonne Δ révèle l'intérêt de l'approche de récupération robuste pour trouver de meilleures solutions pour le problème. La qualité des solutions de l'approche robuste avec récupération est due au fait que celle ci s'adapte au scénario réalisé grâce à la structure de groupes d'opérations permutables.

8.2.2 Résultats sur le problème intégré d'ordonnancement et distribution

Dans cette section, nous évaluons les performances des algorithmes tabou TSR et TOSR sur les instances précédentes. Ces algorithmes sont testés sur des instances de taille variant de 10 à 100 où les paramètres α, β et γ sont fixés à 1. De même que pour les tests de la section précédente effectués sur le problème d'ordonnancement, le paramètre de variation ω prend sa valeur dans l'ensemble $\{0,2,\ 0,4,\ 0,6\}$. Le nombre de scénarios est fixé à $|\mathcal{S}^{'}|$ = 50 et pour chaque couple (n, ω), 10 instances sont générées où $n \in \{10, 20, 30, 50, 100\}$.

La résolution exacte des problèmes d'ordonnancement et de tournées de véhicule robuste par l'intermédiaire des formulations SRM et OSRM proposées (voir sections 6.2 et7.3) étant limitées à des instances de très petites tailles, nous omettons de présenter les résultats obtenus dans cette section. Cependant, les instances résolues de façon exacte sur les instances de taille $n = 10$ retournent des valeurs de l'objectif égales à celles retournées par les heuristiques tabou. A noter que dans l'heuristique TOSR, un batch est constitué d'un seul groupe de livraisons permutables et par conséquent les tests ont été réalisés sur une formulation adaptée à l'heuristique tabou. La résolution exacte d'instance de taille raisonnable étant impossible, les résultats présentés dans la suite de cette section concernent uniquement les heuristiques tabou TSR et TOSR.

La comparaison des deux algorithmes est effectuée en deux phases appelées "*phase d'apprentissage*" et "*phase de validation*". Durant la phase d'apprentissage, nous définissons un ensemble de scénarios $\mathcal{S} \subset \mathcal{S}^{'}$ qui constitue l'ensemble de scénarios d'apprentissage. Ainsi, la comparaison se fait dans un premier temps sur l'ensemble de scénario $\mathcal{S}$ entre les algorithmes TSR et GSR de la version robuste standard et les algorithmes GOSR et TOSR de la version intégrant la flexibilité par l'intermédiaire des groupes de tâches permutables.

La seconde phase qui est celle de la validation consiste à évaluer la meilleure solution obtenue durant la phase d'apprentissage sur l'ensemble de scénarios $\mathcal{S}'$. Ainsi, la meilleure séquence de production, partition des tâches en batch et la meilleure séquence de livraison obtenues durant la première phase par les algorithmes GSR et TSR sont retenues. De la même manière, pour la version flexible de l'algorithme, la meilleure séquence de production de groupes et la meilleure partition des tâches en batchs obtenues durant la phase d'apprentissage par les algorithmes GOSR et TOSR sont retenues. En d'autres termes, la phase d'apprentissage permet de trouver des solutions pour un nombre restreint de scénarios $\mathcal{S} \subset \mathcal{S}'$ et la phase de validation consiste à évaluer les solutions obtenues sur l'ensemble de scénarios $\mathcal{S}'$. Cette façon de faire permet d'évaluer les solutions sur des scénarios non anticipés et donner ainsi des informations supplémentaires sur la robustesse des méthodes.

Les tests sont exécutés sur une machine Intel i3 de 2.40GHz avec 3GO de RAM.

8.2.2.1 Comparaisons sur l'ensemble de scénarios d'apprentissage $\mathcal{S}$

Les algorithmes tabou TSR et TOSR ainsi que les algorithmes gloutons GSR et GOSR sont testés sur les instances présentées plus haut où seuls $|\mathcal{S}|$ scénarios sur un ensemble de $|\mathcal{S}'|$ sont pris en compte. Soit $|\mathcal{S}| \in \{2, 5, 10\}$ l'ensemble de scénarios considérés. Les paramètres de l'algorithme tabou δ ainsi que la taille des listes tabou sont fixés respectivement à 0,4 et au nombre de tâches de l'instance n.

Les algorithmes gloutons et tabou sont testés sur 10 instances de chaque triplet $(n, \mathcal{S}, \omega)$ et les résultats obtenus sont présentés dans la tableau 8.2. Ce premier tableau fournit des résultats agrégés pour chaque triplet $(n, \mathcal{S}, \omega)$ et se divise en trois blocs. Le premier bloc concerne les résultats obtenus par les approches robustes (GSR et TSR) et le second bloc ceux obtenus par les approches GOST et TOSR. Le dernier bloc donne les résultats obtenus en comparant les deux approches. Les colonnes *ttb* indiquent le temps moyen mis par l'algorithme sur les dix instances de chaque triplet $(n, \mathcal{S}, \omega)$ pour atteindre la meilleure solution. Ainsi, le temps moyen mis par l'heuristique tabou TSR est indiqué par la colonne *ttb* du premier bloc et celui mis par GOSR est indiqué par la colonne *ttb* du second bloc. Soit $z(\mathcal{A})$ la valeur de la fonction objectif (i.e., le retard maximum sur l'ensemble de scénarios $\mathcal{S}$) retournée par l'algorithme $\mathcal{A} \in$ {TSR, TOSR, GSR, GOSR} sur une instance particulière. Les colonnes Δ rendent l'écart moyen sur les instances de $(n, \mathcal{S}, \omega)$ entre les solutions retournées par les algorithmes gloutons (solutions initiales) et les heuristiques tabou. Δ est alors égal à $(z(GSR) - z(TSR))/z(GSR)$ dans le premier bloc et à $(z(GOSR) - z(TOSR))/z(GOSR)$ dans le second. Le dernier bloc retourne les écarts moyens entre les solutions initiales des deux approches (GSR et GOSR) et les écarts moyens entre les solutions des algorithmes tabou (TSR et TOSR).Ainsi, $\Delta^1 = (z(GSR) - z(GOSR))/z(GSR)$ et $\Delta^2 = (z(TSR) - z(TOSR))/z(TSR)$.

Nous passons à présent à l'analyse des résultats du tableau 8.2. Premièrement, la colonne Δ^1 révèle que l'algorithme glouton GOSR de l'approche avec récupération on-line donne des résultats nettement supérieurs à ceux retournés par l'algorithme glouton GSR de l'approche robuste standard. Ce premier résultat était attendu vu que le niveau d'optimisation de GSR est très faible comparé celui de GOSR qui s'adapte au scénario réalisé par l'intermédiaire de la structure de groupes. Les deux premiers blocs indiquent que les algorithmes tabou TSR et TOSR améliorent de manière significative les performances des

algorithmes gloutons GSR et GOSR respectivement. Il est intéressant de constater également que les deux algorithmes tabou ont un temps d'exécution comparable sur les mêmes instances et ceci implique que le fait de considérer un niveau d'optimisation supplémentaire (i.e., la création de groupes de tâches permutables) n'affecte pas de manière significative le temps d'exécution. Enfin, les colonnes Δ^2 montrent que dans la plupart des cas, l'approche réactive au scénario réalisé donne de meilleurs résultats que l'approche robuste standard. Ceci étant, nous pouvons voir que sur certaines instances, la qualité des solutions retournées par l'approche robuste est légèrement supérieure. Cette dernière remarque est due au fait que l'optimisation des tournées dans l'approche robuste standard peut induire des solutions meilleures.

8.2.2.2 Évaluation des solutions de la phase d'apprentissage sur l'ensemble de scénarios $\mathcal{S}'$

Dans cette partie de validation, nous retenons pour chaque instance la meilleure solution retournée par les algorithmes durant la phase d'apprentissage. Ainsi dans l'approche robuste, les solutions retenues consistent en une séquence de production des tâches, une partition des tâches en batchs et une séquence de livraison pour chaque batch notées x^{GSR} et x^{TSR} pour l'algorithme glouton GSR et l'heuristique tabou TSR respectivement. D'autre part, les solutions retenues pour les approches réactives GOSR et TOSR consistent en une partition des tâches en groupes d'opérations permutables pour la production, une séquence de production de groupes ainsi que d'une partition des tâches en batchs notées respectivement x^{GOSR} et x^{TOSR}. Les solutions retenues sont alors évaluées sur l'ensemble des scénarios $\mathcal{S}'$.

Les résultats présentés dans le tableau 8.3 sont également divisés en trois blocs. Le premier bloc concerne les résultats obtenus en appliquant les solutions x^{GSR} et x^{TSR}) sur l'ensemble de scénarios $\mathcal{S}'$. Suivant le même raisonnement, le deuxième bloc montre les résultats obtenus en appliquant x^{GOSR} et x^{TOSR} aux $|\mathcal{S}'|$ scénarios. La comparaison des résultats obtenues par les deux approches est donnée dans le dernier bloc.

Soient $z(x) = \max_{s\in\mathcal{S}'}\{L_{\max}(x,s)\}$ avec $x \in \{x^{\mathrm{GSR}}, x^{\mathrm{TSR}}\}$ le retard maximum sur l'ensemble de scénarios $\mathcal{S}'$ obtenu en appliquant les solutions $x \in \{x^{\mathrm{GSR}}, x^{\mathrm{GOSR}}\}$ et soient $z^A(x) = \max_{s\in\mathcal{S}'}\{L_{\max}(A(x,s),s)\}$ le retard maximum sur l'ensemble de scénarios $\mathcal{S}'$ obtenue en appliquant les solutions $x \in \{x^{\mathrm{GOSR}}, x^{\mathrm{TOSR}}\}$ où, $A(x,s)$ est l'algorithme de récupération en ligne qui s'adapte au scénario réalisé.

Les colonnes $\Delta^{'}$ donnent l'écart moyen entre la solution initiale et la solution de l'algorithme tabou. $\Delta^{'}$ est alors égal à.

$(z(x^{\mathrm{GSR}}) - z(x^{\mathrm{TSR}}))/z(x^{\mathrm{GSR}})$ dans le premier bloc et,
$(z^A(x^{\mathrm{GOSR}}) - z^A(x^{\mathrm{TOSR}}))/z^A(x^{\mathrm{GOSR}})$ dans le deuxième bloc.

Cependant, le fait de comparer la solution du pire cas global sur $\mathcal{S}'$ des deux méthodes a un intérêt limité. En effet, il est plus intéressant de comparer les solutions trouvées pour un même scénario permettant d'obtenir des informations sur la qualité des méthodes sur un scénario inexploré. Ainsi, les colonnes $\bar{\Delta}^{'}$ donnent la moyenne de cette comparaison où

$\bar{\Delta}^{'} = moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(x^{\mathrm{GSR}},s) - L_{\max}(x^{\mathrm{TSR}},s)}{L_{\max}(x^{\mathrm{GSR}},s)}$ dans le cas de l'approche robuste standard,

			TSR		TOSR		TSR Vs TOSR	
n	$\mathcal{S}$	ω	$ttb(s)$	$\Delta(\%)$	$ttb(s)$	$\Delta(\%)$	$\Delta^1(\%)$	$\Delta^2(\%)$
10	2	20	0.01	50.42	0.08	23.02	38.31	5.58
		40	0.01	50.09	3.67	23.01	36.08	1.95
		60	0.01	51.39	0.20	26.98	28.75	-4.32
	5	20	7.79	50.27	0.18	22.78	35.77	1.30
		40	0.29	41.71	10.36	18.65	29.67	0.54
		60	0.05	33.46	0.13	17.32	27.46	10.24
	10	20	0.04	48.41	3.02	17.74	38.26	2.40
		40	0.08	41.12	0.64	16.31	31.19	-0.24
		60	0.21	37.97	8.47	13.74	26.85	-0.83
20	2	20	2.70	70.48	17.76	32.90	56.38	2.44
		40	16.07	61.49	8.74	27.47	47.68	1.32
		60	19.33	61.67	6.30	22.15	51.43	3.78
	5	20	3.68	65.69	2.01	28.20	54.60	3.92
		40	9.52	56.03	10.87	21.66	47.08	4.72
		60	10.45	56.69	9.51	17.87	47.79	2.00
	10	20	28.82	62.76	14.32	30.32	48.98	4.88
		40	30.38	53.34	16.68	20.46	45.03	5.62
		60	23.60	50.09	5.55	14.81	44.12	5.01
30	2	20	17.36	68.89	9.96	27.44	55.82	0.88
		40	30.93	64.96	11.63	22.54	55.30	4.50
		60	12.51	67.87	6.78	26.11	57.20	1.44
	5	20	25.88	66.36	27.59	28.28	54.56	3.59
		40	22.26	58.97	7.51	16.81	52.19	4.22
		60	42.83	59.24	24.24	22.11	49.20	3.43
	10	20	15.28	63.25	22.79	23.42	54.50	5.01
		40	25.35	56.45	16.90	16.47	48.80	2.49
		60	21.26	54.39	15.54	20.25	45.54	5.46
50	2	20	24.47	71.33	2.49	21.29	64.22	2.31
		40	25.45	67.42	5.26	21.25	59.17	3.10
		60	14.56	65.27	10.02	18.55	59.16	4.59
	5	20	30.13	68.33	16.19	24.15	59.45	4.27
		40	24.16	61.62	27.54	19.81	54.49	5.26
		60	32.86	56.53	31.75	16.09	51.00	5.95
	10	20	50.60	65.60	32.79	21.16	59.24	6.76
		40	58.49	59.56	56.41	21.99	49.48	3.19
		60	47.13	54.79	55.31	18.01	48.37	6.44
100	2	20	52.73	70.49	21.22	21.14	62.70	1.30
		40	59.31	64.46	25.72	14.83	60.54	5.41
		60	56.97	62.26	9.65	8.61	61.09	4.90
	5	20	109.51	65.84	64.50	15.02	61.21	4.39
		40	110.73	60.82	66.68	20.45	52.08	3.13
		60	113.04	54.41	64.40	13.80	49.11	3.77
	10	20	119.02	58.36	116.47	16.91	58.37	16.15
		40	120.63	55.47	117.90	16.01	51.64	9.28
		60	122.56	44.55	127.80	13.58	47.90	17.82

TABLE 8.2 – Comparaison des méthodes GSR, GOSR, TSR et TOSR sur $|\mathcal{S}| \in \{2, 5, 10\}$

$\bar{\Delta}' = moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(A(x^{\mathrm{GOSR}},s),s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{GOSR}},s),s)}$ dans l'approche robuste avec récupération on-line

La comparaison entre les deux approches est donnée dans le troisième bloc où, $\Delta'^{(1)}$ et $\bar{\Delta'}^{(1)}$ concernent les comparaisons des solutions initiales et $\Delta'^{(2)}$ et $\bar{\Delta'}^{(2)}$ concernent les solutions des algorithmes tabou. En particulier, les colonnes $\Delta'^{(1)}$ et $\bar{\Delta'}^{(1)}$ donnent respectivement l'écart entre les solutions du pire cas sur l'ensemble $\mathcal{S}'$ et l'écart moyen sur tous les scénarios. Les colonnes $\Delta'^{(2)}$ et $\bar{\Delta'}^{(2)}$ restantes indiquent la même information pour les algorithmes tabou cette fois.

$$\Delta'^{(1)} = \frac{z(x^{\mathrm{GSR}}) - z^{A}(x^{\mathrm{GOSR}})}{z(x^{\mathrm{GSR}})}$$

$$\bar{\Delta'}^{(1)} = moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(x^{\mathrm{GSR}},s) - L_{\max}(A(x^{\mathrm{GOSR}},s),s)}{L_{\max}(x^{\mathrm{GSR}},s)}$$

$$\Delta'^{(2)} = \frac{z(x^{\mathrm{TSR}}) - z(x^{\mathrm{TOSR}})}{z(x^{\mathrm{TSR}})}$$

$$\bar{\Delta'}^{(2)} = moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(x^{\mathrm{TSR}},s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(x^{\mathrm{TSR}},s)}$$

Il apparaît clairement d'après les résultats que les solutions issues des approches robustes avec récupération on-line (algorithmes tabou et glouton) sont nettement plus robustes que les solutions issues des approches robustes standards et ceci est vrai pour le pire cas (colonnes $\Delta'^{(1)}$ et $\Delta'^{(2)}$) et la moyenne (colonnes $\bar{\Delta'}^{(1)}$ et $\bar{\Delta'}^{(2)}$). Cet autre résultat montre encore l'aptitude de la structure de groupes à s'adapter au scénario réalisé en utilisant un simple algorithme on-line. Pour ce qui est des deux premiers blocs maintenant, nous pouvons voir que l'algorithme tabou de l'approche robuste TSR améliore significativement la solution initiale GSR. Cependant, le deuxième bloc et en particulier la colonne $\bar{\Delta}'$ montre que sur l'ensemble de scénarios $\mathcal{S}'$, les solutions retournées par l'algorithme tabou TOSR et celles retournés par GOSR sont de qualité similaire.

Dans le but de donner une explication au comportement des solutions de la phase d'apprentissage sur l'ensemble de scénarios $\mathcal{S}'$, nous présentons un dernier tableau de résultats (Tableau. 8.4) dans lequel les résultats du tableau 8.3 sont agrégés. Le tableau ci-dessous est divisé en deux blocs et présente des résultats agrégés de comparaison par rapport à l'algorithme TOSR. Ainsi, les colonnes $\bar{\Delta''}^{(1)}$ and $\bar{\Delta''}^{(4)}$ comparent GSR et TOSR en calculant pour chaque instance l'écart moyen comme suit.

$$moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(x^{\mathrm{GSR}},s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

de même ; les colonnes $\bar{\Delta''}^{(2)}$ et $\bar{\Delta''}^{(5)}$ comparent GOSR et TOSR pour chaque instance comme suit.

$$moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(A(x^{\mathrm{GOSR}},s),s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

et $\bar{\Delta''}^{(3)}$ et $\bar{\Delta''}^{(6)}$ comparent TSR à TOSR également comme suit.

$$moyenne_{s\in\mathcal{S}'} \frac{L_{\max}(x^{\mathrm{TSR}},s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

Pour un ensemble de scénarios $|\mathcal{S}| \in \{2,4,6\}$, le premier bloc (colonnes $\bar{\Delta''}^{(1)}$, $\bar{\Delta''}^{(2)}$ et $\bar{\Delta''}^{(3)}$) donne les écarts par rapport aux moyennes sur tous les paramètres de variation ω

			TSR		TOSR					
n	$\mathcal{S}$	ω	$\Delta^{'}$ (%)	$\bar{\Delta}'$(%)	$\Delta^{'}$ (%)	$\bar{\Delta}'$(%)	$\Delta'^{(1)}$ (%)	$\bar{\Delta'^{(1)}}$(%)	$\Delta'^{(2)}$ (%)	$\bar{\Delta'^{(2)}}$(%)
10	2	20	53.99	36.37	37.42	15.56	50.03	34.69	35.88	12.89
		40	49.35	26.34	40.53	8.30	55.33	28.24	42.86	10.49
		60	47.78	10.73	30.93	-14.09	55.62	27.11	40.89	8.47
	5	20	55.98	40.31	36.21	15.40	51.43	34.96	34.08	8.25
		40	47.46	25.38	36.45	5.89	53.29	30.89	46.62	13.23
		60	39.25	15.26	30.33	-3.77	57.44	29.74	48.83	14.70
	10	20	56.12	42.73	32.25	14.31	52.75	38.37	33.48	9.17
		40	47.07	26.70	39.62	7.31	55.84	31.12	42.55	12.37
		60	49.28	19.73	31.81	1.98	54.60	25.72	39.89	9.30
20	2	20	67.61	52.20	44.61	11.73	67.49	54.92	45.08	18.23
		40	58.68	34.08	32.44	-7.66	65.05	44.36	38.60	9.29
		60	56.85	19.26	26.49	-15.45	62.94	41.24	43.17	16.03
	5	20	68.71	57.24	44.09	14.99	69.62	55.96	40.04	12.70
		40	58.26	39.59	34.45	2.95	65.65	46.85	39.01	14.98
		60	56.95	28.89	30.77	-5.46	65.22	43.61	48.35	17.14
	10	20	67.40	56.96	49.67	22.72	65.79	51.67	40.52	15.22
		40	59.96	42.66	35.04	7.53	66.14	46.46	37.81	13.55
		60	57.03	32.52	30.87	-2.82	62.97	42.93	42.73	13.47
30	2	20	66.30	50.19	33.11	7.12	66.32	55.24	40.95	17.78
		40	58.51	30.80	24.84	-7.24	67.15	48.65	42.33	20.37
		60	60.52	22.25	30.36	-8.97	64.62	43.61	46.64	19.86
	5	20	65.05	53.31	37.01	13.06	68.00	57.21	42.80	19.35
		40	56.70	37.07	27.30	-4.03	69.09	51.82	48.82	21.10
		60	58.24	31.79	27.42	-6.89	65.59	46.74	41.99	17.25
	10	20	64.18	54.10	38.08	14.82	69.08	57.60	44.02	20.33
		40	63.57	45.65	23.76	0.78	68.53	51.69	36.88	12.01
		60	59.58	40.37	33.58	0.84	64.36	45.96	37.97	11.74
50	2	20	64.40	49.50	25.11	4.77	71.88	59.75	43.24	23.48
		40	58.81	31.21	23.70	-2.59	66.27	50.80	48.29	27.06
		60	60.60	22.45	20.20	-6.07	64.60	46.10	49.41	25.98
	5	20	67.36	56.95	33.67	10.06	69.23	59.45	36.70	16.21
		40	58.30	41.19	27.58	1.14	66.47	53.02	45.87	21.72
		60	54.23	31.08	22.62	-4.53	61.08	46.84	39.43	19.08
	10	20	67.76	57.26	30.26	12.48	71.84	60.63	37.61	19.14
		40	62.54	47.83	32.06	8.17	65.45	52.15	37.96	16.15
		60	60.94	41.35	29.50	3.70	63.69	48.43	39.29	15.37
100	2	20	64.35	49.01	22.47	5.25	69.63	60.24	42.98	26.06
		40	59.71	30.93	20.29	1.74	64.07	50.36	46.21	28.27
		60	55.11	15.57	10.77	-1.99	61.98	42.67	46.88	29.74
	5	20	64.12	53.99	17.42	-1.81	70.40	61.88	33.71	15.39
		40	58.39	40.51	21.64	-4.07	63.62	52.98	35.73	17.55
		60	52.19	28.37	16.35	-6.00	61.15	47.53	41.00	21.53
	10	20	58.99	51.58	20.92	3.64	69.68	61.48	37.94	23.04
		40	55.29	43.53	20.31	2.26	64.74	53.56	38.69	19.43
		60	44.24	30.29	16.22	-1.64	59.96	49.21	42.44	25.33

TABLE 8.3 – Validation de la robustesse des solutions de GSR, GOST, TSR et TOSR sur l'ensemble de scénarios $\mathcal{S}'$

sur le scénario $\mathcal{S}$ et pour tout $\omega \in \{0,2,0,4,0,6\}$ le second bloc (colonnes $\bar{\Delta''^{(4)}}$ et $\bar{\Delta''^{(5)}}$) présente les écarts par rapport aux moyennes sur le nombre de scénarios considérés dans la phase d'apprentissage sur ω.

n	$\mathcal{S}$	$\bar{\Delta''^{(1)}}(\%)$	$\bar{\Delta''^{(2)}}(\%)$	$\bar{\Delta''^{(3)}}(\%)$	ω	$\bar{\Delta''^{(4)}}(\%)$	$\bar{\Delta''^{(5)}}(\%)$	$\bar{\Delta''^{(6)}}(\%)$
10	2	63.90	10.13	16.43	20	95.77	21.89	14.34
20		106.61	3.42	23.40		193.07	27.76	25.68
30		110.08	2.82	29.07		179.02	18.63	28.28
50		120.51	1.11	38.31		187.30	13.24	28.61
100		120.05	2.74	42.43		172.49	3.75	30.63
10	5	69.23	11.37	18.33	40	68.10	12.83	20.05
20		128.59	10.81	22.72		98.96	5.25	18.32
30		129.54	5.52	28.77		110.32	0.06	26.66
50		133.52	5.82	26.72		123.56	5.35	31.45
100		120.44	-2.36	24.93		115.30	1.35	31.23
10	10	72.94	12.95	16.35	60	42.20	-0.27	16.72
20		129.86	15.52	20.50		73.04	-3.27	22.62
30		136.92	9.61	20.90		87.21	-0.74	23.79
50		149.37	11.65	23.63		92.53	-0.01	28.59
100		132.97	2.67	32.42		85.68	-2.05	37.92

TABLE 8.4 – Résultats de comparaisons agrégées sur l'ensemble de scénarios $\mathcal{S}'$

Les figures 8.1 et 8.2 synthétisent les résultats du tableau 8.4 où, la couleur bleu représente les colonnes $\bar{\Delta''^{(1)}}$ et $\bar{\Delta''^{(4)}}$, la couleur rouge représente les colonnes $\bar{\Delta''^{(2)}}$ et $\bar{\Delta''^{(5)}}$ et la couleur verte les colonnes $\bar{\Delta''^{(3)}}$ et $\bar{\Delta''^{(6)}}$. De haut en bas, la figure 8.1 présente les résultats pour $s = 2$, $s = 5$ et $s = 10$. De la même manière, la figure 8.2, montre les résultats pour $\omega = 20\%$, $\omega = 40\%$ et $\omega = 60\%$.

Le tableau 8.4 ainsi que les figures 8.1 et 8.2 montrent dans un premier temps que l'algorithme glouton de l'approche robuste standard GSR est de qualité inférieure aux autres algorithmes. Ces résultats montrent également que les performances de l'algorithme tabou de l'approche robuste avec récupération on-line TOSR est de qualité supérieure et relativement stable par rapport à celui de l'approche robuste standard TSR. Cependant, les performances de TOSR comparés à celles de GOSR sont relativement stables par rapport au nombre de scénarios utilisés pendant la phase d'apprentissage mais décroissent par rapport au paramètre de variation ω et au nombre de tâches. Dans certain cas, la qualité de GOSR est même supérieure à celle de TOSR sur $\mathcal{S}'$ et ceci est du au fait que lorsque le degré d'incertitude est tres important, l'intérêt d'optimiser la séquence de groupes devient peu pertinent.

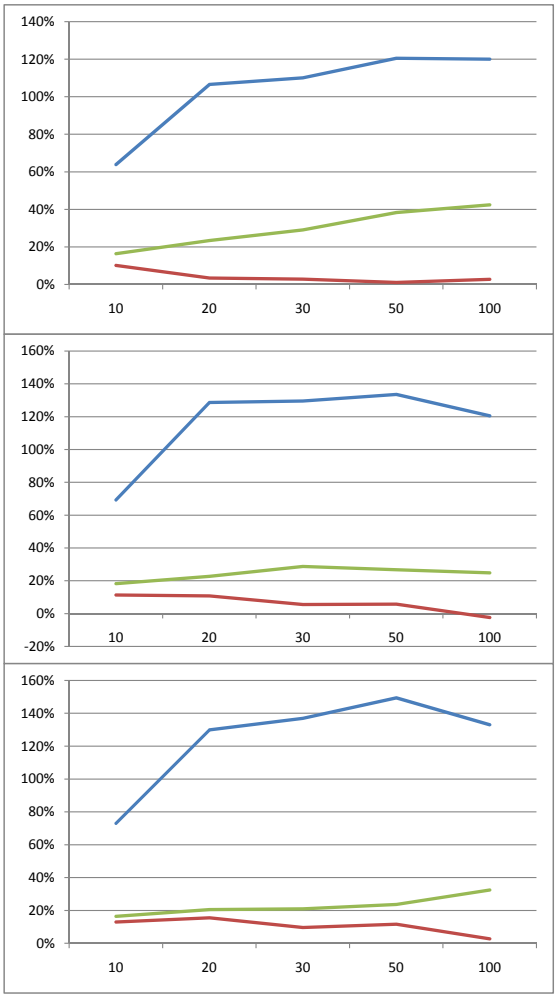


FIGURE 8.1 – Comparaisons pour $|\mathcal{S}| \in \{2, 4, 6\}$ (de haut en bas)

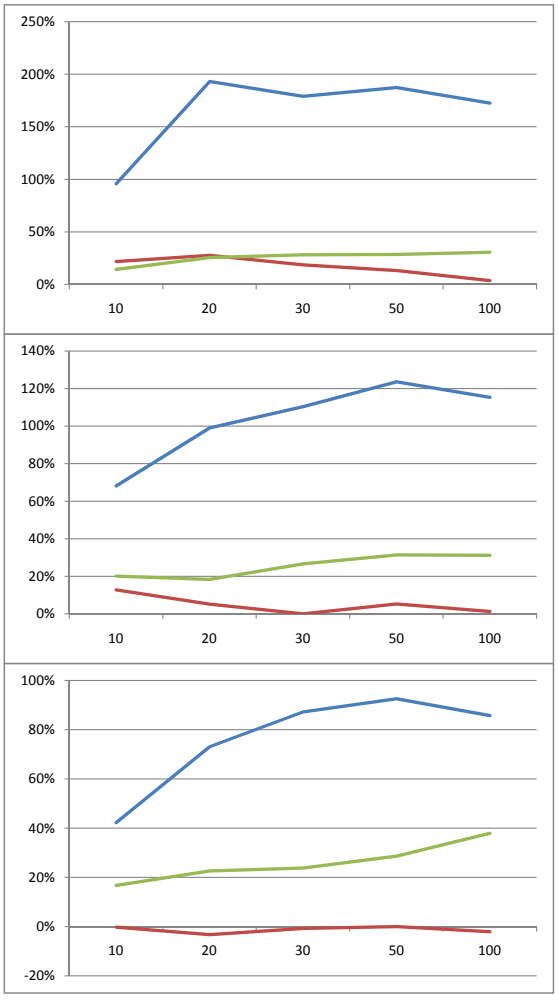


FIGURE 8.2 – Comparaisons pour $\omega \in \{0,2,0,4,0,6\}$ (de haut en bas)

8.3 Conclusion

Dans ce chapitre, nous avons défini la façon dont les instances aléatoires ont été générées. Les méthodes exactes et les méthodes approchées ont été testées pour le problème d'ordonnancement seul, et les méthodes approchées ont été testées pour le problème intégré, en incluant des tests en phase d'apprentissage sur des scénarios connus a priori et ensuite sur des scénarios connus a posteriori. Pour le problème d'ordonnancement seul, les résultats mettent en évidence l'intérêt des groupes de tâches permutables pour l'approche de robustesse avec récupération on-line. Pour le problème intégré, les résultats montrent également l'intérêt des groupes. Toutefois, les résultats sur les scénarios connus a posteriori ne permettent pas d'affirmer la domination de la méthode tabou par rapport à la solution initiale avec groupes de tâches permutables.

Chapitre 9

Conclusion générale

La production et la livraison constituent deux éléments essentiels de la chaîne logistique. Ces problèmes sont généralement traités de façon séparée. Nous proposons dans cette thèse d'aborder de façon conjointe le problème de l'ordonnancement de la production, et celui de la distribution des produits aux clients. Nous considérons exclusivement le cas où la production est réalisée sur une machine unique. Pour la distribution, nous considérons plusieurs cas, selon qu'il n'y a qu'un seule véhicule ou plusieurs, à capacité limitée ou non, et selon qu'il y a un unique site à livrer ou plusieurs. La fiabilité des données est un point essentiel en planification, aussi nous abordons le cas où les données sont incertaines, et connues sous forme de scénarios, et nous recherchons des solutions robustes.

Après une introduction générale qui présente les problèmes abordés et un état de l'art, la thèse se compose de deux parties.

Dans la première partie, nous considérons que les données sont connues de façon certaine (cas déterministe).

Nous abordons dans un premier temps le problème en considérant que la séquence de production et la séquence de distribution sont identiques, et que cette séquence est connue. Le problème consiste alors uniquement à constituer les batchs de livraison. Nous montrons que ce problème est NP-difficile et nous proposons un algorithme de programmation dynamique pseudo-polynomial pour le résoudre. Des cas particuliers de ce problème ont également été étudiés, et pour chacun d'eux, une preuve de NP-complétude ou un algorithme de programmation dynamique polynomial sont proposés. Pour le cas "séquence fixée", il nous semble qu'un dernier cas particulier mérite d'être abordé dans le futur. Il s'agit du cas où les durées des tâches sont toutes identiques.

Nous abordons ensuite le cas où la livraison est sous-traitée à un transporteur. Les dates de départ des véhicules sont fixées, les véhicules ont une capacité limitée (volume), et il n'y a qu'un seul site à livrer. Nous cherchons à minimiser la somme des dates de livraison, critère peu abordé dans la littérature pour ce problème. Nous montrons la difficulté du problème et nous proposons une heuristique d'approximation pour un cas particulier. Pour le problème général nous proposons une formulation compacte et une formulation étendue adaptée à la génération de colonnes. Pour ce problème, une perspective consiste à développer un algorithme de Branch& Price pour valider l'intérêt de la génération de colonnes en résolvant

le problème de façon exacte.

Nous abordons enfin le cas général où les tâches doivent être livrées à plusieurs clients par un unique véhicule. Deux fonctions objectif sont considérées : la plus grande date de fin de livraison et la somme des dates de livraison. Le séquencement de la production, la composition des batchs et la construction des tournées du véhicule constituent donc le problème à résoudre. Après une brève étude de complexité de quelques cas particuliers, nous proposons des formulations adaptées pour la génération de colonnes. La génération de colonnes fournit pour le problème relâché des bornes inférieures de très bonne qualité pour le critère makespan jusqu'à 100 tâches. Les perspectives pour ce chapitre consistent à développer un algorithme de Branch & Price pour obtenir les solutions exactes, mais aussi des méthodes approchées de type tabou, pour avoir des éléments de comparaison et résoudre des instances de plus grande taille.

Dans la seconde partie, nous considérons que les données ne sont pas connues de façon certaine (scénarios) et nous cherchons des solutions robustes. Le critère considéré dans cette partie est la minimisation du plus grand retard algébrique dans le pire des cas (on cherche une solution robuste la meilleure possible pour tous les scénarios).

Nous commençons par présenter la robustesse, un état de l'art, puis la notion de groupes de tâches permutables.

Nous proposons en un premier temps d'aborder le problème avec une approche "standard", proposée par Kouvelis et Yu. Nous proposons plusieurs modèles de programmation mathématique et un algorithme tabou pour le problème d'ordonnancement seul et pour le problème intégré. Les perspectives de ce travail consistent à améliorer les heuristiques, mais aussi à proposer des méthodes exactes de décomposition.

Nous proposons en un deuxième temps une nouvelle manière d'utiliser le concept de groupes de tâches permutables pour une approche robuste "avec récupération on-line". La séquence de groupes de production et de livraison sont les mêmes pour tous les scénarios. La séquence d'ordonnancement des tâches à l'intérieur de chaque groupe de production et la séquence de livraison des tâches à l'intérieur de chaque groupe de livraison ne sont pas fixées a priori. L'originalité de la méthode réside dans l'utilisation de cette flexibilité pour construire une solution finale à l'aide d'un algorithme glouton on-line qui s'adapte au scénario réalisé.

Nous proposons des modèles linéaires en nombres entiers, des algorithmes gloutons ainsi qu'une méthode tabou pour le problème d'ordonnancement seul et pour le problème intégré.

Afin de comparer cette nouvelle approche aux approches robustes standards, des instances de tailles raisonnables ont été générées aléatoirement et la robustesse des approches a été comparée. Dans chacun des cas, les résultats expérimentaux montrent que pour un temps d'exécution comparable, les résultats des approches robustes avec récupération on-line sont supérieurs à ceux des approches robustes standard. Cependant, l'intérêt des groupes de tâches permutables semble décroître lorsque le niveau d'incertitude est élevé. Maintenant que l'intérêt des groupes de tâches permutables a été établi pour trouver des solutions robustes, une perspective consiste à mettre au point des algorithmes encore plus performants, qui implémentent les groupes.

Ce travail a été réalisé grâce au financement du projet ANR ATHENA numéro ANR-13-BS02-0006-01.

Bibliographie

[Agra *et al.*, 2013] Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M. et Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40(3):856–866.

[Aissi *et al.*, 2009] Aissi, H., Bazgan, C. et Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems : A survey. *European journal of operational research*, 197(2):427–438.

[Aloulou et Della Croce, 2008] Aloulou, M. A. et Della Croce, F. (2008). Complexity of single machine scheduling problems under scenario-based uncertainty. *Operations Research Letters*, 36(3):338–342.

[Aloulou *et al.*, 2004] Aloulou, M. A., Kovalyov, M. Y. et Portmann, M.-C. (2004). Maximization problems in single machine scheduling. *Annals of Operations Research*, 129(1-4):21–32.

[Armstrong *et al.*, 2008] Armstrong, R., Gao, S. et Lei, L. (2008). A zero-inventory production and distribution problem with a fixed customer sequence. *Annals of Operations Research*, 159(1):395–414.

[Artigues *et al.*, 2016] Artigues, C., Billaut, J.-C., Cheref, A., Mebarki, N. et Yahouni, Z. (2016). *Robust Machine Scheduling Based on Group of Permutable Jobs*, pages 191–220. Springer International Publishing, Cham.

[Artigues *et al.*, 2005a] Artigues, C., Billaut, J.-C. et Esswein, C. (2005a). Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2):314–328.

[Artigues *et al.*, 2005b] Artigues, C., Billaut, J.-C. et Esswein, C. (2005b). Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research*, 165(2):314–328.

[Aytug *et al.*, 2005] Aytug, H., Lawley, M. A., McKay, K., Mohan, S. et Uzsoy, R. (2005). Executing production schedules in the face of uncertainties : A review and some future directions. *European Journal of Operational Research*, 161(1):86–110.

[Azi *et al.*, 2007] Azi, N., Gendreau, M. et Potvin, J.-Y. (2007). An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European journal of operational research*, 178(3):755–766.

[Azi *et al.*, 2010] Azi, N., Gendreau, M. et Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763.

[Ben-Tal *et al.*, 2004] Ben-Tal, A., Goryashko, A., Guslitzer, E. et Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

[Berlińska et Drozdowski, 2011] Berlińska, J. et Drozdowski, M. (2011). Scheduling divisible mapreduce computations. *Journal of Parallel and Distributed Computing*, 71(3): 450–459.

[Berlińska et Drozdowski, 2015] Berlińska, J. et Drozdowski, M. (2015). Scheduling multilayer divisible computations. *RAIRO-Operations Research*, 49(2):339–368.

[Bertsimas et Sim, 2003] Bertsimas, D. et Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical programming*, 98(1):49–71.

[Bertsimas et Sim, 2004] Bertsimas, D. et Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35–53.

[Bertsimas et Simchi-Levi, 1996] Bertsimas, D. J. et Simchi-Levi, D. (1996). A new generation of vehicle routing research : robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304.

[Billaut *et al.*, 2008] Billaut, J.-C., Moukrim, A. et Sanlaville, E. (2008). *Flexibility and robustness in scheduling*. John Wiley & Sons.

[Billaut et Roubellat, 1996] Billaut, J.-C. et Roubellat, F. (1996). A new method for workshop real time scheduling. *International Journal of Production Research*, 34(6): 1555–1579.

[Błażewicz *et al.*, 2015] Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G. et Weglarz, J. (2015). *Handbook on scheduling : from theory to applications*. Springer Science & Business Media.

[Brucker, 2007] Brucker, P. (2007). *Scheduling Algorithms. Fifth edition.* Springer-Verlag Berlin Heidelberg.

[Caprara *et al.*, 2014] Caprara, A., Galli, L., Stiller, S. et Toth, P. (2014). Delay-robust event scheduling. *Operations Research*, 62(2):274–283.

[Cattaruzza *et al.*, 2016] Cattaruzza, D., Absi, N. et Feillet, D. (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2):676–693.

[Chang et Lee, 2004] Chang, Y.-C. et Lee, C.-Y. (2004). Machine scheduling with job delivery coordination. *European Journal of Operational Research*, 158(2):470–487.

[Chen, 2010a] Chen, J.-S. (2010a). Integration of job scheduling with delivery vehicles routing. *Information technology journal*, 9(6):1202–1206.

[Chen, 2010b] Chen, Z.-L. (2010b). Integrated production and outbound distribution scheduling : review and extensions. *Operations research*, 58(1):130–148.

[Chen et Pundoor, 2006] Chen, Z.-L. et Pundoor, G. (2006). Order assignment and scheduling in a supply chain. *Operations Research*, 54(3):555–572.

[Chen et Pundoor, 2009] Chen, Z.-L. et Pundoor, G. (2009). Integrated order scheduling and packing. *Production and Operations Management*, 18(6):672–692.

[Chen et Vairaktarakis, 2005] Chen, Z.-L. et Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science*, 51(4):614–628.

[Cheref *et al.*, 2016a] Cheref, A., Agnetis, A., Artigues, C. et Billaut, J.-C. (2016a). Fixed-sequence single machine scheduling and outbound delivery problems. *In Proceedings of 5th the International Conference on Operations Research and Enterprise Systems*, pages 145–151.

[Cheref *et al.*, 2016b] Cheref, A., Artigues, C. et Billaut, J.-C. (2016b). A new robust approach for a production scheduling and delivery routing problem. *IFAC-PapersOnLine*, 49(12):886–891.

[Cheref *et al.*, 2016c] Cheref, A., Artigues, C. et Billaut, J.-C. (2016c). Online recoverable robustness based on groups of permutable jobs for integrated production scheduling and delivery routing. Rapport technique.

[Cheref *et al.*, 2016d] Cheref, A., Artigues, C., Billaut, J.-C. et Ngueveu, S. U. (2016d). Integrated production scheduling and delivery routing : complexity results and column generation. *In International Symposium on Combinatorial Optimization*, pages 439–450. Springer.

[Cheref *et al.*, 2014] Cheref, A., Bouchard, T., Billaut, J.-C. et Artigues, C. (2014). Un algorithme tabou pour résoudre, grâce aux groupes d'opérations permutables, un problème d'ordonnancement et de routing robuste. *In Conf érence Internationale de MOd élisation, Optimisation et SIMulation-MOSIM'14*.

[Christensen *et al.*, 2017] Christensen, H. I., Khan, A., Pokutta, S. et Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing : A survey. *Computer Science Review*.

[Coffman Jr *et al.*, 2013] Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S. et Vigo, D. (2013). Bin packing approximation algorithms : survey and classification. *In Handbook of Combinatorial Optimization*, pages 455–531. Springer.

[Condotta *et al.*, 2013] Condotta, A., Knust, S., Meier, D. et Shakhlevich, N. V. (2013). Tabu search and lower bounds for a combined production–transportation problem. *Computers & Operations Research*, 40(3):886–900.

[Daniels et Kouvelis, 1995a] Daniels, R. L. et Kouvelis, P. (1995a). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):363–376.

[Daniels et Kouvelis, 1995b] Daniels, R. L. et Kouvelis, P. (1995b). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):363–376.

[Devapriya *et al.*, 2016] Devapriya, P., Ferrell, W. et Geismar, N. (2016). Integrated production and distribution scheduling with a perishable product. *European Journal of Operational Research*.

[Dong *et al.*, 2013] Dong, J., Zhang, A., Chen, Y. et Yang, Q. (2013). Approximation algorithms for two-machine open shop scheduling with batch and delivery coordination. *Theoretical Computer Science*, 491:94–102.

[Dósa et Sgall, 2013] Dósa, G. et Sgall, J. (2013). First fit bin packing : A tight analysis. *In LIPIcs-Leibniz International Proceedings in Informatics*, volume 20. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Dósa et Sgall, 2014] Dósa, G. et Sgall, J. (2014). Optimal analysis of best fit bin packing. *In International Colloquium on Automata, Languages, and Programming*, pages 429–441. Springer.

[Dror, 1994] Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978.

[Feillet *et al.*, 2004] Feillet, D., Dejax, P., Gendreau, M. et Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems. *Networks*, 44(3):216–229.

[Gabrel *et al.*, 2014] Gabrel, V., Murat, C. et Thiele, A. (2014). Recent advances in robust optimization : An overview. *European journal of operational research*, 235(3):471–483.

[Gao *et al.*, 2015] Gao, S., Qi, L. et Lei, L. (2015). Integrated batch production and distribution scheduling with limited vehicle capacity. *International Journal of Production Economics*, 160:13–25.

[Garey et Johnson, 1975] Garey, M. R. et Johnson, D. S. (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411.

[Garey et Johnson, 1979] Garey, M. R. et Johnson, D. S. (1979). Computers and intractability : A guide to the theory of np-completeness.

[Garey *et al.*, 1976] Garey, M. R., Johnson, D. S. et Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129.

[Garey *et al.*, 1988] Garey, M. R., Tarjan, R. E. et Wilfong, G. T. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13(2):330–348.

[Geismar *et al.*, 2008] Geismar, H. N., Laporte, G., Lei, L. et Sriskandarajah, C. (2008). The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing*, 20(1):21–33.

[Gendreau *et al.*, 1996] Gendreau, M., Laporte, G. et Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12.

[Glover, 1989] Glover, F. (1989). Tabu search ?part i. *ORSA Journal on computing*, 1(3):190–206.

[Gounaris *et al.*, 2013] Gounaris, C. E., Wiesemann, W. et Floudas, C. A. (2013). The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, 61(3):677–693.

[Graham *et al.*, 1979] Graham, R. L., Lawler, E. L., Lenstra, J. K. et Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of discrete mathematics*, 5:287–326.

[Hall *et al.*, 2001] Hall, N. G., Lesaoana, M. et Potts, C. N. (2001). Scheduling with fixed delivery dates. *Operations Research*, 49(1):134–144.

[He *et al.*, 2006] He, Y., Zhong, W. et Gu, H. (2006). Improved algorithms for two single machine scheduling problems. *Theoretical Computer Science*, 363(3):257–265.

[Hernandez *et al.*, 2016] Hernandez, F., Feillet, D., Giroudeau, R. et Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, 249(2):551–559.

[Herroelen et Leus, 2004] Herroelen, W. et Leus, R. (2004). Robust and reactive project scheduling : a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620.

[Herroelen et Leus, 2005] Herroelen, W. et Leus, R. (2005). Project scheduling under uncertainty : Survey and research potentials. *European journal of operational research*, 165(2):289–306.

[Johnson, 1954] Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics (NRL)*, 1(1):61–68.

[Karaoğlan et Kesen, 2017] Karaoğlan, İ. et Kesen, S. E. (2017). The coordinated production and transportation scheduling problem with a time-sensitive product : a branch-and-cut algorithm. *International Journal of Production Research*, 55(2):536–557.

[Kellerer et Kotov, 2003] Kellerer, H. et Kotov, V. (2003). An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing. *Operations Research Letters*, 31(1):35–41.

[Kouvelis et Yu, 1997] Kouvelis, P. et Yu, G. (1997). *Robust Discrete Optimization and Its Applications*. Springer US.

[Lee et Chen, 2001] Lee, C.-Y. et Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, 4(1):3–24.

[Lee, 2015] Lee, I. S. (2015). A coordinated scheduling of production-and-delivery under dynamic delivery cost environments. *Computers & Industrial Engineering*, 81:22–35.

[Lee *et al.*, 2014] Lee, J., Kim, B.-I., Johnson, A. L. et Lee, K. (2014). The nuclear medicine production and delivery problem. *European Journal of Operational Research*, 236(2):461–472.

[Lenstra *et al.*, 1977] Lenstra, J. K., Kan, A. R. et Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of discrete mathematics*, 1:343–362.

[Li et Ou, 2005] Li, C.-L. et Ou, J. (2005). Machine scheduling with pickup and delivery. *Naval Research Logistics (NRL)*, 52(7):617–630.

[Li et Vairaktarakis, 2007] Li, C.-L. et Vairaktarakis, G. (2007). Coordinating production and distribution of jobs with bundling operations. *IIE transactions*, 39(2):203–215.

[Li *et al.*, 2005] Li, C.-L., Vairaktarakis, G. et Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research*, 164(1):39–51.

[Liebchen *et al.*, 2009] Liebchen, C., Lübbecke, M., Möhring, R. et Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. *In Robust and online large-scale optimization*, pages 1–27. Springer.

[Lopez et Roubellat, 2008] Lopez, P. et Roubellat, F. (2008). *Production Scheduling*. Wiley-ISTE.

[Lozano *et al.*, 2015] Lozano, L., Duque, D. et Medaglia, A. L. (2015). An exact algorithm for the elementary shortest path problem with resource constraints. *Transportation Science*, 50(1):348–357.

[Lu *et al.*, 2008] Lu, L., Yuan, J. et Zhang, L. (2008). Single machine scheduling with release dates and job delivery to minimize the makespan. *Theoretical Computer Science*, 393(1-3):102–108.

[Lysgaard et Wøhlk, 2014] Lysgaard, J. et Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research*, 236(3):800–810.

[Macedo *et al.*, 2011] Macedo, R., Alves, C., de Carvalho, J. V., Clautiaux, F. et Hanafi, S. (2011). Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. *European Journal of Operational Research*, 214(3):536–545.

[Moons *et al.*, 2016] Moons, S., Ramaekers, K., Caris, A. et Arda, Y. (2016). Integrating production scheduling and vehicle routing decisions at the operational decision level : a review and discussion. *Computers & Industrial Engineering*.

[NEO, 2017] NEO (2017). Vehicle routing problem description. `http://neo.lcc.uma.es/vrp/vehicle-routing-problem/`. Accessed : 2017-03-15.

[Scholz-Reiter *et al.*, 2010] Scholz-Reiter, B., Frazzon, E. M. et Makuschewitz, T. (2010). Integrating manufacturing and logistic systems along global supply chains. *CIRP Journal of Manufacturing Science and Technology*, 2(3):216–223.

[Simchi-Levi, 1994] Simchi-Levi, D. (1994). New worst-case results for the bin-packing problem. *Naval Research Logistics (NRL)*, 41(4):579–585.

[Spieksma, 1994] Spieksma, F. C. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers & operations research*, 21(1):19–25.

[Toth et Vigo, 2014] Toth, P. et Vigo, D. (2014). *Vehicle routing : problems, methods, and applications*. SIAM.

[Ullrich, 2013] Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research*, 227(1):152–165.

[Verderame *et al.*, 2010] Verderame, P. M., Elia, J. A., Li, J. et Floudas, C. A. (2010). Planning and scheduling under uncertainty : a review across multiple sectors. *Industrial & engineering chemistry research*, 49(9):3993–4017.

[Viergutz et Knust, 2014] Viergutz, C. et Knust, S. (2014). Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research*, 213(1): 293–318.

[Wang et Cheng, 2009] Wang, X. et Cheng, T. E. (2009). Production scheduling with supply and delivery considerations to minimize the makespan. *European journal of operational research*, 194(3):743–752.

[Wu *et al.*, 1999] Wu, S. D., Byeon, E.-S. et Storer, R. H. (1999). A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124.

[Zhong *et al.*, 2007] Zhong, W., Dósa, G. et Tan, Z. (2007). On the machine scheduling problem with job delivery coordination. *European Journal of Operational Research*, 182(3):1057–1072.

Résumé :

La production et la distribution sont deux éléments essentiels dans une chaîne de production. En effet, dans de nombreux systèmes de production, les produits finis sont livrés de l'usine vers différents clients, entrepôts ou centres de distribution. Afin d'assurer une optimisation globale des performances, nous abordons dans cette thèse le problème intégré d'ordonnancement et de distribution. Dans la littérature, de nombreux articles traitent des approches intégrées, impliquant des décisions de production et de distribution à un niveau stratégique. De plus en plus d'articles abordent ces problèmes au niveau opérationnel. De nouveaux problèmes de cette catégorie sont abordés dans cette thèse. Dans un premier temps le problème déterministe est étudié, et plusieurs cas sont abordés. Pour chacun d'eux, une étude de complexité est proposée et des algorithmes de résolution exacte (programmation dynamique, programmation mathématique, génération de colonnes) sont proposés, ainsi que des méthodes approchées. L'incertitude sur les données est introduite dans la deuxième partie de la thèse et des méthodes exactes et heuristiques sont proposées pour résoudre des problèmes intégrés robustes d'ordonnancement et de distribution. En particulier, des approches de programmation linéaire en nombres entiers et des métaheuristiques de type tabou sont proposées pour résoudre les problèmes d'optimisation robuste standard et d'optimisation "robuste récupérable". Ces méthodes sont évaluées et les résultats montrent l'efficacité des méthodes que nous proposons.

Mots clés : Recherche opérationnelle, ordonnancement, tournées de véhicules, complexité, programmation dynamique, génération de colonnes, robustesse.

Abstract :

Production and delivery are two essential elements in supply chain management. Indeed, in many production systems, finished products are delivered from the factory to different customers, warehouses or distribution centers. In order to ensure a global optimization of performances, we address in this thesis the integrated problem of scheduling and delivery. In the literature, a large number of papers deal with integrated approaches at a strategic level. More and more papers deal with these problems at an operational level. New problems in this category are addressed in this thesis. In a first part, the deterministic problem is studied and several cases are treated. For each of them, a study of complexity is proposed and exact (dynamic programming, mathematical programming, column generation) as well as approximated methods are proposed. Data uncertainty is introduced in the second part of the thesis and exact and heuristic methods are proposed to solve integrated problems of scheduling and delivery. In particular, integer linear programming approaches and tabu search heuristic are proposed to solve the standard robust approach and the "recoverable robustness" approach. These methods are evaluated and the results show the efficiency of the methods that we propose.

Keywords : Operations research, scheduling, vehicle routing, complexity, dynamic programming, column generation, robustness.