Titre: Title:	Modèles et méthodes pour l'optimisation du processus de préparation de commandes dans les entrepôts de type e-commerce
Auteur: Author:	Mustapha Haouassi
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Haouassi, M. (2022). Modèles et méthodes pour l'optimisation du processus de préparation de commandes dans les entrepôts de type e-commerce [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10693/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10693/
Directeurs de recherche: Advisors:	Louis-Martin Rousseau, & Jean-Charles Billaut
Programme: Program:	Doctorat en génie industriel

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

ET

UNIVERSITÉ DE TOURS

Modèles et méthodes pour l'optimisation du processus de préparation de commandes dans les entrepôts de type e-commerce

MUSTAPHA HAOUASSI
Département de mathématiques et de génie industriel
Polytechnique Montréal
et
Département d'informatique
Université de Tours

Thèse en cotutelle présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie industriel

Juillet 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

ET

UNIVERSITÉ DE TOURS

Cette thèse intitulée :

Modèles et méthodes pour l'optimisation du processus de préparation de commandes dans les entrepôts de type e-commerce

présentée par **Mustapha HAOUASSI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Nadia LAHRICHI, présidente
Louis-Martin ROUSSEAU, membre et directeur de recherche
Jean-Charles BILLAUT, membre et codirecteur de recherche
Jean-François CORDEAU, membre
Christelle GUERET, membre externe

REMERCIEMENTS

Je souhaite tout d'abord exprimer ma profonde gratitude envers mes directeurs de recherche : Louis-Martin Rousseau, Jorge Mendoza et Yannick Kergosien. Leur expertise dans de nombreux domaines a été d'un grand apport pour mener les différents projets de recherche. Je les remercie aussi pour tout le soutien qu'ils m'ont apporté sur le plan humain. J'ai bien conscience que l'aboutissement de cette thèse leur revient certainement en premier lieu.

Faire une thèse en cotutelle m'a permis de découvrir deux environnements différents, à savoir l'équipe ROOT de l'université de Tours et le laboratoire du Cirrelt à Montréal. Dans les deux cas, j'ai us la chance d'évoluer avec des personnes riches en valeurs humaines. Je tiens donc à remercier les différentes personnes que j'ai connues durant cette thèse pour tous les bons moments qu'on a partagés ensemble, que ce soit dans les laboratoires ou en dehors. Merci aussi aux professeurs Nadia Lahrichi, Jean-CharlesBillaut, Jean-François Cordeau, Richard Goureau et Christelle Gueret d'avoir accepté d'être membres de mon jury de thèse et d'évaluer mes travaux. Une mention particulière pour le professeur Jean-François Cordeau pour les différentes remarques pertinentes qui m'ont permis d'améliorer considérablement mon manuscrit de thèse.

Je tiens aussi à remercier ma famille pour l'amour et le soutien qu'ils m'ont apportés durant cette thèse, à commencer par mes parents Hayette et Djeloul ainsi que mon frère Sid-Ali et sa petite famille. Je n'oublie pas aussi ma femme Assia, qui m'a apporté un soutien inconditionnel durant les périodes difficiles de cette thèse et qui a même contribué dans la rédaction du manuscrit. Et pour finir, je tiens aussi à remercier mon fils Aylane-Younes, né durant cette thèse, qui m'a fait découvrir le privilège d'être un papa et m'a apporté une incroyable énergie positive me permettant de surmonter les derniers obstacles de cette thèse.

Je voudrais finir par avoir une grande pensée pour ma grande mère Khadija, décédée durant la première année de cette thèse. Ce fut une personne incroyable qui m'a tant appris dans la vie. Je lui dédie l'ensemble de ce travail !

RÉSUMÉ

Dans les dernières décennies, les activités liées au e-commerce ont connu un développement considérable à travers le monde, donnant lieu à un environnement extrêmement compétitif. Pour s'approprier des parts de marché, les entreprises proposent des services de plus en plus innovants comme la livraison rapide. La mise en place de ce type de services passe par une bonne gestion de la chaine logistique, et notamment la politique de stockage et de préparation des commandes à l'intérieur des entrepôts. Ces politiques doivent s'adapter à un nouveau pattern de demande caractérisé par un grand nombre de commandes à préparer quotidiennement, chacune composée de peu d'items. De plus, les offres de livraison rapide font que chaque commande arrive avec une date limite de préparation qu'il est impératif de respecter.

Certains entrepôts d'e-commerce implémentent des systèmes automatisés dans lesquels la préparation des commandes est principalement prise en charge par des robots. Cependant, ces systèmes engendrent des coûts opératoires extrêmement élevés, souvent difficiles à assumer pour des entreprises de petite et moyenne tailles. La majorité des entrepôts suivent encore des systèmes classiques dans lesquels des préparateurs de commandes démarrent d'un dépôt central, se déplacent à pied dans les allées de l'entrepôt pour récupérer les différents items des commandes et les ramènent à l'unité d'assemblage. Afin de s'adapter au pattern de demande associée au e-commerce, ces entrepôts implémentent de nouvelles techniques de stockage et de préparation des commandes à l'échelle tactique et opérationnelle. L'objectif de cette thèse est de proposer des modèles et des méthodes pour optimiser la préparation des commandes dans les entrepôts implémentant ces techniques et d'étudier leur impact.

Dans la première partie de cette thèse, nous étudions l'impact d'une pratique émergente dans les entrepôts d'e-commerce, à savoir la possibilité de prendre en charge une commande par plusieurs préparateurs (la collecte des items d'une commande est répartie sur plusieurs personnes). Nous considérons un ensemble de demandes avec des dates limites de préparation et un ensemble de préparateurs munis de chariots à capacité limitée. L'objectif du premier problème étudié est de planifier les tournées de chaque préparateur afin de collecter tous les items des commandes à temps et de minimiser la durée totale de collecte. La résolution de ce problème nécessite de déterminer le regroupement en lots des items, l'affectation des lots aux préparateurs et l'itinéraire de collecte des items de chaque lot. Pour résoudre le problème, nous proposons une heuristique basée sur le principe "route-first schedule-second". Dans la phase de

routage, l'heuristique répartit les items de toutes les commandes en grands lots et construit les tournées associées à chaque lot en utilisant une adaptation de la procédure de *Split*. Dans la phase d'ordonnancement, les tournées construites sont affectées aux préparateurs (et séquencées) à l'aide de la programmation par contraintes. À travers une comparaison utilisant un benchmark d'instances publiques, de notre méthode avec une méthode de la littérature qui résout la version du problème où l'ensemble des items d'une même commande doit être collecté par un même préparateur, nous démontrons qu'une réduction de la durée totale de collecte de 30 % en moyenne peut être réalisée en autorisant la collecte d'items d'une commande par plusieurs préparateurs.

Dans la deuxième partie de la thèse, nous étudions l'adaptation du problème de routage d'un préparateur (c.-à-d. définir la séquence de visite des différents emplacements de stockage lors d'une tournée afin de minimiser la distance parcourue) dans un nouveau type d'entrepôt, dit à étagères mixtes (mélangées). Contrairement aux entrepôts classiques où tout le stock associé à un item est disponible dans une seule étagère, les entrepôts à étagères mixtes divisent le stock disponible d'un item en petites unités qui sont dispersées dans différentes étagères un peu partout dans l'entrepôt. La résolution de ce problème passe par la sélection des emplacements d'où récupérer les différents items avant de construire la tournée de collecte. Pour résoudre le problème, nous proposons une méthode exacte de décomposition de type "logic-based Benders" dans laquelle la sélection des emplacements de stockage se fait dans le problème maître et la construction de la tournée qui visite les emplacements sélectionnés dans le sous-problème. Nous proposons une coupe d'optimalité dont nous prouvons la validité. De plus, nous introduisons un ensemble d'améliorations, incluant une fonction dite de borne inférieure et des inégalités valides, qui jouent un rôle important dans l'efficacité de la méthode. Finalement, une comparaison avec une adaptation d'une méthode exacte, initialement proposée pour un problème de routage classique, au contexte des entrepôts à étagères mixtes démontre l'efficacité de notre méthode.

Dans la dernière partie de cette thèse, nous étudions la combinaison de deux pratiques, à savoir le stockage à étagères mixtes et le découpage de l'espace de ramassage en plusieurs zones disjointes avec un seul préparateur opérant dans chaque zone. Nous considérons un problème de planification de collecte d'items de commandes par vague, où chaque préparateur effectue une tournée de collecte. L'objectif du problème est de sélectionner d'où récupérer chaque item et de définir la tournée associée à chaque préparateur de telle sorte à minimiser le temps de collecte (la plus longue tournée parmi tous les préparateurs). Pour résoudre le problème, nous proposons une adaptation de la méthode introduite dans la deuxième partie

de la thèse et une simple adaptation d'une méthode de la littérature spécialisée pour une structure particulière d'entrepôts (1-bloc). Nous démontrons dans la partie expérimentale que l'implémentation d'une stratégie de zonage dans les entrepôts à étagères mixtes réduit considérablement les temps de collecte par rapport au zonage dans les entrepôts classiques (sans étagères mixtes).

ABSTRACT

In the last decades, e-commerce activities have grown significantly throughout the world, leading to a highly competitive environment. To gain market share, companies are offering innovative services such as fast delivery. Implementing these services requires good management of the supply chain, more precisely, the storage and order picking policies inside the warehouses. These policies must be adapted to a new demand pattern characterized by a large number of orders to be prepared daily, each composed of a few items. Moreover, the orders come with deadlines which must be respected to ensure high service quality.

To face these new requirements, some e-commerce warehouses implement automated systems in which order picking is mainly performed by robots. However, these systems generate high operating costs that are often difficult to handle for small and medium-sized companies. Most warehouses still follow the classic picker-to-part system in which order pickers start from a central depot, walk through the warehouse aisles to collect items contained in picklists, and bring them to the assembly unit. In order to adapt to the demand pattern of e-commerce, new storage and picking techniques are implemented in these warehouses at the tactical and operational levels. The general purpose of this thesis is to propose models and methods that optimize the order picking process in warehouses using these techniques and to study their impact.

In the first part of this thesis, we study the impact of an emerging practice in e-commerce warehouses, i.e., the possibility of handling an order by several pickers (splitting an order). To this end, we assume a set of orders with preparation deadlines that must be handled by a set of pickers equipped with capacitated carts. We generalize the integrated orders batching, batch scheduling, and picker routing problem by allowing the orders splitting. To solve the problem, we propose a route-first schedule-second heuristic. In the routing phase, the heuristic divides the orders into clusters and designs the tours that retrieve the items of each cluster using a split-based procedure. In the scheduling phase, the constructed tours are assigned to pickers (and sequenced) using constraint programming. On a publicly available benchmark, we compare our results against a state-of-the-art iterated local search algorithm designed for the non-splitting order version of the problem. We show that splitting the orders using our algorithm reduces the picking time by 30% on average.

In the second part of the thesis, we study the adaptation of the picker routing problem (PRP) , *i.e.*, sequencing the storage locations that must be visited in a single tour such that the total travel distance is minimized, into a new type of warehouses, called mixed-shelves warehouses. Unlike conventional warehouses, where all the inventory associated with an item is available on a single shelf, mixed-shelves warehouses break up the total inventory into smaller units that are assigned to different shelves throughout the warehouse. Thus, the locations from where to retrieve the items must be selected before designing the picking tour. To solve the problem, we propose a logic-based Benders decomposition method in which the storage locations are selected in the master problem, and the tour that visits the selected locations is determined in the subproblem. We design tailored optimality cuts and prove their validity. In addition, we introduce a set of algorithmic enhancements, including a lower bounding function and a set of valid inequalities, that significantly improve the method. Finally, a comparison with an adaptation of an exact method, initially proposed for the PRP, to the context of mixed-shelves warehouses demonstrates the superiority of our method.

In the last part of this thesis, we study the combination of two practices, namely mixed shelves storage and zone picking (dividing the picking area into disjoint zones with a picker operating in each zone). In our problem, a wave of orders must be prepared by collecting all the items of each order in the next tour of the pickers. The objective is to select from where to retrieve each item of the orders and to define the tour of each picker such that the makespan is minimized (the longest tour among the pickers). To solve the problem, we propose an adaptation of the method introduced in the second part of the thesis and an adaptation of another method from the literature for a particular warehouse shape (1-block). We demonstrate in the experimental part that implementing a zoning strategy in mixed-shelves warehouses significantly reduces the makespan compared to zoning in conventional warehouses.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Avec l'émergence de la globalisation et le développement rapide des technologies de l'information et des communications (TIC), le monde a connu dans les dernières décennies un changement radical dans les habitudes d'achat et l'apparition d'un nouveau modèle de vente, à savoir le commerce électronique (e-commerce) [3]. Au Canada, le revenu total des détaillants d'e-commerce est passé de 21.9 milliards de dollars en 2017 à 25.3 milliards de dollars en 2019 entrainant ainsi une augmentation de 15.5 % dans cette période.

Cette tendance s'est fortement accélérée avec l'escalade de la pandémie de COVID-19. En effet, la rapide propagation du virus à travers le monde a contraint les gouvernements de plusieurs pays à temporairement fermer leurs frontières et à imposer différentes mesures sociales et économiques (télétravail, couvre-feu, distanciation sociale, réduction du nombre de travailleurs, fermeture des commerces non essentiels, etc.). Ces mesures ont poussé les clients à se tourner vers les plateformes en ligne pour effectuer leurs achats. Les ventes en ligne chez Walmart-USA, par exemple, ont connu une augmentation de 74 % au cours de la pandémie. Au Canada, le total des ventes en e-commerce est passé à 32.4 milliards de dollars en 2021. La tendance risque de durer, voire même d'augmenter, avec des prévisions aux alentours de 40 milliards de dollars en 2025 [4].

Tous ces changements ont créé un environnement hautement compétitif entre commerçants avec des clients de plus en plus exigeants. Pour se faire une place dans un tel marché, les compagnies se doivent de proposer des services de plus en plus innovants. Parmi ces services figure "la livraison rapide". En effet, certaines compagnies offrent à leurs clients la possibilité de les livrer en des délais préférentiels. Amazon par exemple, à travers son service *PrimeNow*, propose à ses abonnés de Manhattan, de Paris ou encore de Londres de livrer les commandes passées en ligne en 1h (avec un coût de livraison supplémentaire), en 2h ou pendant la journée gratuitement. Les entreprises offrant ce genre de services reçoivent quotidiennement un grand nombre de commandes (demandes) urgentes avec très peu de produits (items) par demande. Le nombre moyen d'items d'une commande d'Amazon en Allemagne par exemple est 1.8 [5]. De plus, ces entreprises doivent proposer un catalogue fourni de produits pour cibler une clientèle aussi large que possible.

Afin de s'adapter à ces nouvelles offres, il est nécessaire de mettre en place un bon système de gestion des différentes opérations de la chaine logistique. L'entreposage, c'est-à-dire

le stockage temporaire de produits entre deux étages successifs d'une chaine logistique, figure parmi les opérations les plus importantes à gérer [6]. Il fait globalement intervenir les processus suivants :

- Réception : qui englobe toutes les tâches liées au déchargement, à la vérification de la correspondance entre les quantités demandées et les quantités reçues, à l'inspection de l'état des différentes palettes de commandes et potentiellement la réorganisation des palettes avant de les introduire dans l'entrepôt.
- Stockage : qui consiste à mettre les produits reçus dans les différentes étagères (emplacements de stockage) de l'entrepôt. Pour cela, il faut entre autres lister les différentes étagères vides au moment de la réception de la marchandise et choisir l'étagère associée à chaque produit.
- Préparation de commandes : lors de la réception des différentes commandes, un ensemble d'activités se mettent en place incluant la collecte (le ramassage) des bons items dans l'entrepôt, le triage, l'empaquetage et finalement le transfert des commandes vers l'unité de livraison.
- Expédition : une fois les commandes préparées, elles sont acheminées vers le prochain destinataire de la chaine logistique. Il peut s'agir d'un autre entrepôt, d'un centre de distribution ou d'un consommateur final. Cela se fait par les propres moyens de l'entreprise (camions, vélos, piétons, etc.) ou en passant par une compagnie tierce spécialisée dans l'expédition de produits.

Dans cette thèse, nous allons principalement nous concentrer sur le processus de préparation de commandes. En effet, ce dernier a été identifié par les professionnels comme celui qui impacte le plus la chaine logistique, et donc à mettre au centre des préoccupations [7]. Il peut engendrer jusqu'à 60-70 % des coûts opératoires [8]. Dans leur récente revue de la littérature, Van Gils et al. [9] classifient les décisions liées à la gestion de la préparation des commandes à trois niveaux :

- Stratégique : Il s'agit des différentes décisions relatives à la structure de l'entrepôt, aux différents équipements pour le stockage et la collecte des items et aux systèmes mis en place afin de garantir une compétitivité dans une vision à long terme.
- Tactique : Ce niveau englobe les décisions relatives au stockage des produits dans les étagères de l'entrepôt, au découpage de l'entrepôt en zones, ou encore à l'affectation des préparateurs aux zones par exemple. Ces décisions ont un impact à moyen terme.
- Opérationnel : Ce niveau englobe les différentes tâches et décisions faites au quotidien qui ont un impact en temps réel sur la performance du processus. Parmi ces tâches, on peut citer : le calcul des tournées de collecte, le regroupement des commandes en

lots ou encore l'affectation des lots aux préparateurs.

Évidemment, les décisions au niveau opérationnel sont grandement connectées avec les choix faits aux niveaux tactique et stratégique.

Différents systèmes de préparation des commandes sont implémentés dans les entrepôts d'e-commerce. Ils suivent globalement deux schémas :

- *article vers humain* : Dans un tel système, une flotte de robots se déplace dans l'entrepôt, transporte des étagères mobiles vers l'unité d'assemblage où des opérateurs statiques récupèrent les produits nécessaires pour satisfaire les différentes commandes. Les étagères sont ensuite remises à leur place ou placées à d'autres endroits dans l'entrepôt. Ce type de système, bien qu'il soit très efficace à l'échelle opérationnelle, nécessite un très grand niveau d'automatisation dans l'entrepôt, ce qui engendre des coûts d'investissement, de fonctionnement et de maintenance extrêmement importants. De plus, ils sont difficilement adaptables aux changements dans la charge de travail causés par la période des fêtes ou l'apparition d'événements imprévus [10].
- *humain vers article* : Il s'agit du système le plus répandu en pratique et qui fera l'objet de cette thèse. Par exemple plus de 80% des entrepôts en Europe de l'Ouest suivent ce système [5]. Cette approche se caractérise par une flotte de préparateurs de commandes qui démarrent d'un dépôt central (ou un ensemble de dépôts) avec un chariot, effectuent des tournées dans l'entrepôt pour collecter les différents items qui leur ont été assignés et les ramènent au dépôt ou directement vers l'unité d'assemblage. Les items sont stockés dans des étagères qui forment une disposition en blocs composée d'allées de ramassage et de couloirs transversaux [11]. Les allées de ramassage sont parallèles entre elles et comprennent des étagères disposées face à face des deux côtés de chaque allée. Les couloirs transversaux ne contiennent pas d'étagères, mais permettent aux préparateurs de commandes d'entrer ou de sortir d'une allée de ramassage. De plus, ils divisent l'entrepôt en plusieurs blocs et les allées de ramassage en "sous-allées". Un schéma d'un entrepôt à deux blocs est donné dans la figure 4.2. Au cours d'une journée de travail, un préparateur de commandes effectue plusieurs tournées. Il est guidé par des listes de collecte dans laquelle les différents items à ramasser dans chaque tournée sont énumérés.

Une gestion efficace de l'entreposage passe nécessairement par la mise en place d'une bonne politique de stockage et d'une bonne politique de préparation des commandes. La politique

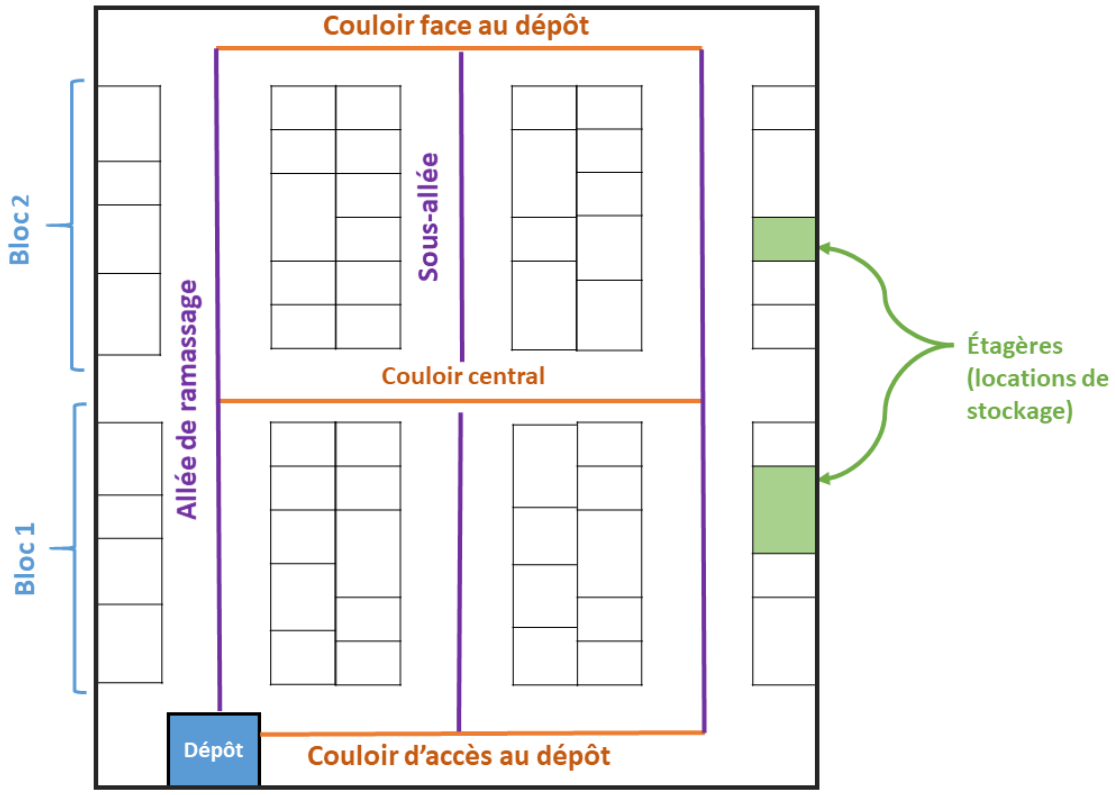


FIGURE 1.1 Disposition en deux blocs

de stockage s'intéresse à la manière de stocker les items dans l'entrepôt *a priori* de telle sorte à faciliter leur ramassage *a postériori*. Un ensemble de problèmes et de modèles sont introduits dans la littérature afin d'optimiser ce processus. Une vue d'ensemble de ces problèmes sera donnée dans la section 2.1 du chapitre 2. De façon générale, les entrepôts traditionnels implémentent des stratégies à travers lesquelles les réserves d'un item sont stockées dans une seule étagère ou dans des étagères collées l'une à l'autre. Ces stratégies, bien qu'elles aient l'avantage de faciliter le repérage des différents items par les préparateurs de commandes, engendrent des tournées de collecte potentiellement très longues. C'est surtout le cas lorsque les demandes sont constituées de très peu d'items, car les préparateurs de commandes ont tendance à parcourir de longues distances dans l'entrepôt sans récupérer des items (improductivité). Pour faire face à cet inconvénient, des entreprises telles qu'Amazon ou Zalando implémentent une stratégie de stockage dite à étagères mixtes (ou mélangées). À travers cette stratégie, les réserves d'un item sont coupées en petites quantités et réparties dans différentes étagères un peu partout dans l'entrepôt. L'idée derrière cette stratégie est d'augmenter la probabilité de trouver des items appartenant à la liste de collecte courante dans une même

allée où dans des allées proches réduisant ainsi le temps d'improductivité [5]. Cette stratégie permet aussi une meilleure utilisation de l'espace de stockage vu que les items peuvent être affectés à n'importe quelle étagère vide qui peut les contenir.

La politique de préparation des commandes s'intéresse à la manière de récupérer les différentes commandes dans l'entrepôt. Au niveau tactique, certains entrepôts divisent l'espace de collecte en plusieurs zones et affectent un (ou plusieurs) préparateur(s) à chaque zone. Ceci permet entre autres de réduire les problèmes de congestion dans une allée entre les préparateurs [7]. Au niveau opérationnel, la planification des tournées de collecte s'organise de différentes façons selon la nature de la demande. Lorsqu'une commande est constituée de plusieurs items, un système de "collecte par commande" est implémenté à travers lequel chaque préparateur de commandes prend en charge une seule commande à la fois. Lorsque les commandes sont de petites tailles, il devient intéressant d'assembler plusieurs commandes en lot pour optimiser le processus ("collecte par lot"). L'objectif à optimiser dans les deux cas est le temps total des différentes tournées. Ce dernier est fonction d'un ensemble d'activités explicitées dans la figure 1.2. Les problèmes qui résultent de ces modes de planification ont été largement étudiés par la communauté scientifique. L'essentiel de ces travaux sera présenté dans le chapitre 2.

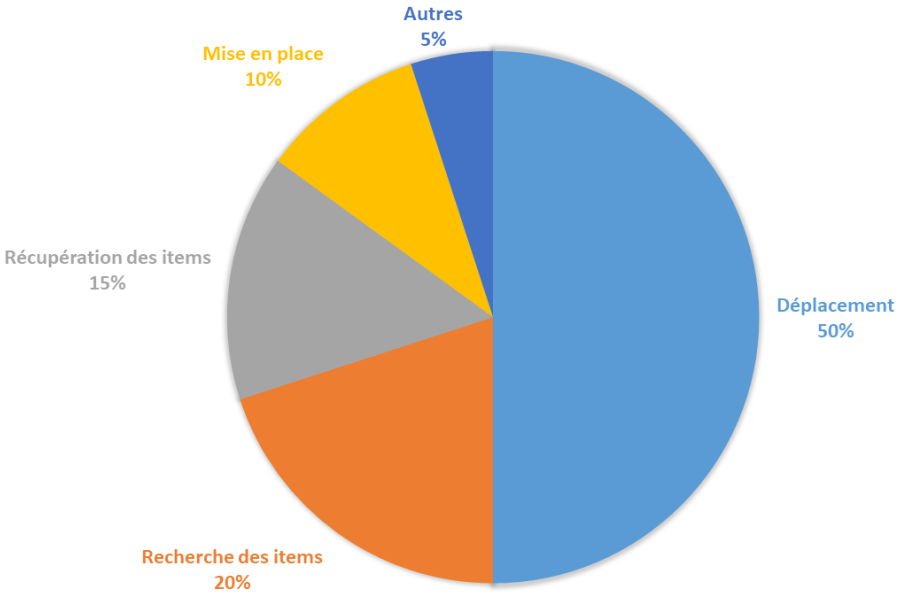


FIGURE 1.2 Pourcentage de temps des différentes activités d'une tournée de collecte [1]

Dans les entrepôts d'e-commerce, les commandes, en plus d'être de petites tailles, arrivent avec des dates limites de préparation à respecter. Le respect de ces dates est primordial pour assurer un niveau de service performant à sa clientèle. En effet, toute insatisfaction peut rapidement engendrer des pertes considérables pour l'entreprise. Par conséquent, il est important d'apporter les politiques adéquates à l'échelle tactique et à l'échelle opérationnelle pour répondre à ces spécificités.

Les travaux de recherche de cette thèse proposent des modèles et des méthodes pour gérer le processus de préparation des commandes en prenant en compte les nouvelles pratiques émergentes dans les entrepôts d'e-commerce. De plus, ces méthodes sont utilisées pour étudier l'impact de ces pratiques sur la performance du processus global. À travers le chapitre 4, la question de connaitre le potentiel bénéfice qui pourrait être obtenu du fait de répartir les items d'une commande sur plusieurs préparateurs sera posée. Cette pratique est très peu considérée dans les modèles proposés dans la littérature. Ceci est dû au fait qu'elle soit peu utilisée en pratique puisqu'elle engendre des efforts supplémentaires pour l'assemblage des commandes. Cependant, les entrepôts d'aujourd'hui disposent de technologies et de systèmes logistiques qui leur permettent de traiter efficacement la consolidation des items en commandes et d'implémenter cette pratique.

Le chapitre 5 se focalise sur comment calculer une tournée d'un préparateur de commandes dans les entrepôts où une stratégie de stockage à étagères mixtes est implémentée. Pour résoudre ce problème, une méthode exacte qui prend en charge des entrepôts partitionnés en plusieurs blocs sera proposée. Finalement, le chapitre 6 se concentre sur la gestion des tournées dans un entrepôt dans lequel l'espace de collecte est découpé en zones avec un préparateur opérant dans chaque zone et les items sont répartis dans les étagères avec une politique de stockage à étagères mixtes. Des méthodes pour organiser le ramassage d'une vague de commandes par les préparateurs seront proposées. Les résultats montreront comment l'implémentation d'une politique de stockage à étagères mixtes peut améliorer la préparation de commandes dans un tel contexte.

La suite de cette thèse sera organisée de la manière suivante. Le chapitre 2 survole l'essentiel des travaux sur l'entreposage dans les entrepôts traditionnels et les entrepôts d'e-commerce. Le chapitre 3 décrit l'organisation générale de la thèse. Le chapitre 4 présente l'article, intitulé *The integrated orderline batching, batch scheduling, and picker routing problem with multiple pickers : the benefits of splitting customer orders* et publié dans le journal *Flexible Services and Manufacturing Journal* le 1 juillet 2021. Le chapitre 5 présente l'article *The picker routing*

problem in mixed-shelves, multi-block warehoues soumis au journal *Computers & Operations Research* le 4 juillet 2022. Le chapitre 6 présente une étude comparative entre la politique de stockage traditionnelle et la politique de stockage à étagères mixtes dans les entrepôts découpés en zones. Le chapitre 7 présente une synthèse des travaux réalisés, leurs limites ainsi que des perspectives de recherche. Finalement, le chapitre 8 apporte une conclusion générale aux travaux de recherche réalisés dans cette thèse.

CHAPITRE 2 REVUE DE LA LITTÉRATURE

Ces dernières années, la littérature scientifique a témoigné un grand intérêt à l'étude de l'entreposage et de la préparation des commandes dans les entrepôts de distribution suivant le système *humain vers article*. Dans ce chapitre, nous allons présenter l'essentiel de ces travaux en quatre sections. La section 2.1 présente l'ensemble des études traitant des problèmes émergents dans le processus de stockage. La section 2.2 est dédiée à le découpage d'entrepôts en zones. La section 2.3 présente une vue d'ensemble sur les travaux concernant le processus de préparation des commandes. Enfin, le positionnement de nos travaux par rapport à la littérature est discuté dans la section 2.4.

2.1 Processus de stockage

D'un point de vue stratégique, certaines entreprises décomposent leurs entrepôts en une zone de réserve et une zone de collecte des items. Des réapprovisionnements de la zone de réserve (zone où on stocke les items pour des périodes de durée assez importante) vers la zone de collecte (zone d'où on ramasse les items pour former les différentes commandes) sont faits périodiquement. La zone de collecte devrait être relativement petite par rapport à la zone de réserve pour minimiser les trajets des préparateurs de commandes. Cependant, une telle configuration engendre des approvisionnements réguliers entre les deux zones et ces approvisionnements doivent généralement se faire dans des périodes d'inactivité dans les zones de collecte. Par conséquent, il y a un compromis à faire entre les espaces alloués à chaque zone et le nombre d'approvisionnements à faire entre les deux zones. Le problème qui traite de ces décisions s'appelle le "forward-reserve problem" [12]. Un autre concept intéressant est celui du stockage dynamique. Dans ce dernier, on maintient une zone de collecte très petite et on transfert les items de la zone de réserve vers la zone de collecte de telle sorte à retrouver les items dans la zone de collecte juste à temps pour pouvoir les collecter (avec des carrousels, une grue automatisée, etc.). Ce concept, bien que présent en pratique, a reçu peu d'intérêt dans la communauté scientifique [13].

Au niveau de la zone de collecte, les entrepôts implémentent différentes stratégies de stockage avec chacune des avantages et des inconvénients. Les critères de mesure de performance de ces stratégies sont généralement la familiarité des préparateurs de commandes avec les emplacements de stockage, la bonne occupation de l'espace de stockage et la réduction du temps de parcours des préparateurs. Ces politiques peuvent se répartir en deux ensembles selon la

disponibilité de prévisions sur la demande future ou non [6]. Lorsque ces informations ne sont pas disponibles, il est difficile de faire mieux que des stratégies simples qui se concentrent principalement sur la gestion et l'ergonomie de l'espace de stockage et des préparateurs de commandes. Parmi ces stratégies, la stratégie dite aléatoire affecte chaque item à un emplacement de stockage parmi ceux qui sont vides d'une manière aléatoire [14]. Ces décisions se font à l'aide d'un système informatisé ou bien en laissant les préparateurs choisir eux-mêmes l'emplacement où mettre chaque item. Dans le deuxième cas, les étagères des entrepôts ont tendance à bien se remplir aux alentours du dépôt (endroit d'où démarrent les préparateurs) et à se vider au fur et à mesure qu'on s'en éloigne [13]. L'avantage de cette stratégie est globalement qu'elle permet une meilleure utilisation de l'espace de stockage. Une alternative est de "dédier" un emplacement de stockage à chaque item de telle sorte que cet emplacement de stockage ne peut recevoir aucun autre item dans tous les réapprovisionnements futurs. Cette technique est utile lorsque l'entrepôt reçoit les mêmes items avec des quantités stables à chaque réapprovisionnement. Son principal avantage est qu'elle permet aux préparateurs de mémoriser l'emplacement associé à chaque item.

Lorsque l'historique sur les commandes passées nous permet de calculer des prévisions sur la demande future, il est intéressant d'exploiter ces informations pour établir une meilleure affectation des items aux emplacements de stockage. L'une des manières de le faire est de calculer des indices qui trient les items selon un ordre de priorité. Le plus connu est le COI ("Cuber-Per-Order Index") introduit par Heskett [15]. Le COI d'un item correspond au ratio entre l'espace qu'il occupe dans l'entrepôt et la fréquence avec laquelle il a été collecté dans une période de temps donnée. L'idée est de dédier les emplacements de stockage les plus intéressants (globalement ceux qui sont les plus proches du dépôt) aux items qui partent le plus rapidement, c'est-à-dire qui sont à la fois les plus demandés et qui occupent le moins d'espace.

Une stratégie intermédiaire entre le stockage aléatoire et le stockage dédié est le stockage par classes. Elle s'implémente en divisant l'ensemble des items à stocker en groupes (trois en général) selon le principe de popularité de Pareto et en affectant les items de chaque groupe à une zone prédéfinie d'une manière aléatoire [16]. Une autre façon de définir les groupes est de calculer des indices d'interactions entre les items (le nombre de fois que deux items apparaissent dans une même commande, par exemple) et de mettre les items qui interagissent le plus dans le même groupe.

Tous les travaux précédents supposent que les copies d'un même item sont stockées dans un seul emplacement de stockage. Or, comme mentionné dans l'introduction, certains entrepôts d'e-commerce implémentent une nouvelle stratégie de stockage dans laquelle un item est éparpillé un peu partout dans l'entrepôt. Dans ce contexte, Weidinger et Boysen [17] se sont intéressés à la façon d'affecter les items aux emplacements de stockage. L'idée de leur étude est de définir dans l'entrepôt un ensemble de repères (qui correspondent globalement à des dépôts intermédiaires) et d'affecter des items aux emplacements de stockage disponibles de telle sorte à minimiser la distance maximale entre les items et les repères prédéfinis. Pour résoudre le problème, ils proposent une heuristique de recherche binaire initialement utilisée pour résoudre le problème de p-centres. Dans la partie expérimentale, ils proposent un ensemble de perspectives managériales sur la fréquence avec laquelle le réapprovisionnement de la zone de collecte en items devrait se faire. Albán et al. [18] définissent un problème dans lequel un ensemble d'items doit être affecté aux emplacements de stockage pour répondre à un ensemble de demandes déjà connues. Dans leur modèle, chaque emplacement de stockage a une capacité limitée et une variable de décision correspond à l'affectation d'un item à un emplacement de stockage pour répondre à une demande donnée. Par conséquent, la quantité de chaque item stockée dans un emplacement doit être suffisante pour répondre à la demande de cet item. Le problème est résolu avec une formulation en PLNE (programme linéaire en nombres entiers) renforcée avec des inégalités valides. Jiang et al. [19] étudient le problème d'affectation des items aux emplacements de stockage dans un centre de distribution. L'idée derrière leur modèle est de définir des corrélations entre les items à partir de l'historique des commandes et de répartir les items dans l'entrepôt de telle sorte à mettre les items les plus corrélés proches les uns des autres. Sachant qu'une stratégie de stockage à étagères mixtes est supposée, la distance entre deux items A et B dans une solution réalisable du problème correspond à la plus courte distance entre les emplacements où est stocké A et les emplacements où est stocké B.

2.2 Découpage de l'entrepôt en zones

En adoptant une stratégie de zonage, l'espace de collecte est coupé en plusieurs zones. Chaque zone est définie par un ensemble d'allées dans lesquelles un ou plusieurs préparateurs de commandes opèrent. Cela permet aux préparateurs de commandes de moins se chevaucher entre eux et de mieux se familiariser avec les items car ils couvrent des espaces plus petits. La littérature identifie deux approches pour le zonage des entrepôts : le zonage séquentiel et le zonage synchronisé [20]. Dans le zonage séquentiel, les zones de l'entrepôt sont classifiées (de la plus proche du dépôt à la plus lointaine, par exemple) et chaque liste de collecte est prise

en charge d'une manière progressive. Ainsi, le premier préparateur récupère et place la partie de la liste identifiée dans sa zone dans des boîtes qu'il transmet au préparateur de la zone suivante. Le processus se poursuit jusqu'à ce que tous les items de la liste soient collectés. Ce système peut être utile dans les entrepôts d'épicerie en ligne par exemple où certaines caractéristiques des items (poids, fragilité, produits congelés...) font que des contraintes de préséance sur les types d'items existent. Le critère pour juger de la performance d'un zonage est généralement la minimisation de la date de fin de collecte d'une liste [21]. Dans le cas d'un zonage séquentiel, il correspond à la somme des distances parcourues dans chaque zone.

Dans le zonage synchronisé, les préparateurs opèrent d'une manière parallèle sur la liste de collecte courante. Chaque préparateur revient au dépôt central avec un sous-ensemble d'items de la liste récupéré dans sa zone ou les dépose dans un dépôt alternatif qui a un accès direct à un système de convoyage. Cette stratégie est particulièrement adaptée à la réalité des entrepôts d'e-commerce où les demandes arrivent avec des dates limites de préparation serrées [22]. La date de fin de la collecte d'une liste dans un système synchronisé correspond à la plus grande distance parcourue par les tours de chaque zone. Ce type de critère permet par ailleurs un équilibrage de la charge de travail entre les différents préparateurs [23]. La littérature sur le zonage séquentiel et le zonage synchronisé s'intéresse globalement à l'étude de l'impact de décisions tactiques telles que la taille du système et le nombre de zones [7, 24] ou encore la combinaison du zonage avec d'autres stratégies de stockage et de regroupement de demandes en lots [25, 26] sur la performance des systèmes. Ces questions sont généralement investiguées en proposant des modèles analytiques appliquant des simulations et des réseaux de files d'attente. Plus récemment, on retrouve certains travaux qui se placent dans un environnement dynamique où la taille de chaque zone peut changer en temps réel (d'une liste de collecte à une autre) et font donc partie des variables décisionnelles. Dans ce contexte, Saylam et al. [23] étudient un problème de préparation des commandes dans un entrepôt implémentant une stratégie de zonage dynamique et synchronisée. Ils proposent un programme dynamique qui résout le problème pour les entrepôts à un bloc.

2.3 Processus de préparation de commandes

Le processus de préparation de commandes a reçu un grand intérêt scientifique pour son importance pratique. La littérature dans ce domaine peut être divisée en trois catégories : dans la première catégorie, le problème étudié consiste à trouver la meilleure tournée possible pour récupérer les items contenus dans une liste de collecte (problème de routage). Dans la deuxième catégorie, il faut d'abord construire les listes d'items en regroupant un ensemble de

demandes en lots avant de définir la tournée de collecte des items de chaque lot (problème de regroupement de demandes en lots). Dans la dernière catégorie, les demandes arrivent avec des dates limites de préparation. Par conséquent, en plus de définir des lots de demandes et une tournée pour chaque lot, il faut aussi affecter et séquencer les tournées sur un ensemble de préparateurs de telle sorte à garantir un certain niveau de service en fonction du respect des dates limites de préparation (problème de regroupement de demandes en lots et de séquençage des lots sur les préparateurs). Dans cette section, la sous-section 2.3.1 présente l'essentiel des travaux sur le problème de routage. La sous-section 2.3.2 expose les travaux relatifs au problème de regroupement de demandes en lots. Enfin, la sous-section 2.3.3 présente les travaux relatifs au problème de regroupement de demandes en lots et de séquençage des lots sur les préparateurs.

2.3.1 Problème de routage

Le problème de routage dans sa version classique peut être énoncé de la manière suivante [27] : étant donné un préparateur de commandes démarrant (et finissant) sa tournée dans un dépôt central et étant donné un ensemble d'emplacements de stockage à visiter pour récupérer une liste d'items, dans quelle séquence les emplacements de stockage doivent-ils être visités pour minimiser la distance totale parcourue par le préparateur ?

Une manière de traiter ce problème est de le modéliser comme un problème de voyageur de commerce (TSP) dans un graphe complet et symétrique où l'ensemble des sommets correspond au dépôt et aux emplacements de stockage. Le coût d'une arrête dans ce graphe correspond à la distance du plus court chemin entre les sommets composant cette arête. Ainsi, l'ensemble des méthodes exactes et heuristiques résolvant le problème du TSP (voir [28]) peuvent être exploitées pour résoudre ce problème.

Une autre classe de méthodes qui prennent en compte la structure de l'entrepôt existe dans la littérature. En effet, les entrepôts suivent globalement une structure rectangulaire impliquant que les préparateurs de commande se déplacent selon la métrique de Manhattan. Par conséquent, un ensemble de propriétés peut être exploité pour développer des méthodes spécialisées au problème de routage dans les entrepôts. Ratliff et Rosenthal [27] ont introduit un théorème fondamental limitant le nombre de possibilités de visiter une allée de ramassage dans une solution optimale à six indépendamment du nombre d'emplacements de stockage à visiter dans cette allée. De plus, ils ont proposé un programme dynamique qui résout le problème pour le cas spécial des entrepôts à un bloc. Leurs travaux ont ensuite été étendus

par Roodbergen et De Koster [11] aux entrepôts à deux blocs. Plus récemment, Pansart et al. [29] ont proposé un ensemble d'améliorations sur deux formulations de flots de l'état de l'art qui résolvent le problème du TSP et du Steiner-TSP (trouver le plus court tour dans un graphe dans lequel certains nœuds peuvent être visités plus d'une fois et d'autres sont optionnels [30]). Ces améliorations incluent des coupes et du prétraitement qui exploitent la structure de l'entrepôt. De plus, ils adaptent un programme dynamique, initialement conçu par Cambazard et Catusse [31] pour le problème du TSP-rectilinéaire (problème de TSP dans lequel la distance entre chaque paire de sommets suit la métrique de Manhattan), au problème de routage d'un préparateur pour un nombre arbitraire de blocs. Les résultats expérimentaux ont mis en avant le bénéfice considérable qu'on peut tirer en exploitant la structure de l'entrepôt pour résoudre ce problème. En effet, les renforcements proposés ont pu fournir une grande amélioration aux formulations de flot basiques. De plus, le programme dynamique a surpassé CONCORDE, le puissant solveur dédié à la résolution du problème de TSP. Finalement, Schiffer et al. [10] proposent un programme dynamique bidirectionnel qui réduit de 67% les temps de calcul du programme dynamique introduit dans [29] et une formulation en PLNE pour résoudre efficacement le problème.

Bien que les méthodes explicitées en haut produisent des solutions optimales, elles ont grandement divisé la communauté scientifique sur leur intérêt d'un point de vue pratique. Certains chercheurs jugent les tournées que produisent ces méthodes contre-intuitives pour les préparateurs de commandes, et par conséquent, difficiles à mettre en place. Au lieu de cela, ils pensent qu'il serait plus judicieux de se tourner vers des politiques de routage plus simples et plus intuitives comme : traverser entièrement chaque allée qui contient un emplacement de stockage à visiter (S-shape) ou encore entrer et sortir de chaque allée qui contient les emplacements de stockage à visiter par le même point (return). D'autres chercheurs mettent en avant le potentiel bénéfice à tirer en implémentant des tournées de collecte optimales. Theys et al. [32] ont démontré que la réduction du temps de parcours pouvait aller jusqu'à 47% avec des solutions optimales par rapport à de simples politiques. De plus, ils mettent en avant le fait que les entrepôts modernes sont équipés de systèmes permettant de bien guider les préparateurs de commandes dans leurs tournées [33].

Dans les entrepôts à étagères mixtes, un autre niveau de décision intervient. En effet, chaque item dans la liste peut se trouver en quantité limitée dans différents emplacements de stockage. Par conséquent, il faut décider à la fois d'où récupérer les différents items et définir la plus courte tournée pour les récupérer. Peu de travaux dans la littérature s'intéressent à résoudre ce problème. Weidinger [34] propose une heuristique en deux étapes pour résoudre

le problème. Dans la première étape, un ensemble de règles de sélection permettent de choisir les emplacements de stockage d'où récupérer les items. Le programme dynamique de [27] est ensuite utilisé dans la deuxième étape pour construire la plus courte tournée qui visite les emplacements sélectionnés. Goeke et Schneider [35] proposent une formulation en PLNE pour résoudre le problème de routage classique qu'ils adaptent ensuite à différentes spécificités des entrepôts de e-commerce dont le stockage à étagères mixtes. La formulation exploite les caractéristiques d'une solution optimale introduites dans [27]. Cependant, elle se spécialise sur les entrepôts à un bloc seulement et n'est pas extensible à des structures à plusieurs blocs.

2.3.2 Problème de regroupement des demandes en lots

Un nombre important d'articles dans la littérature étudient l'optimisation du processus de préparation des commandes dans un contexte où les demandes sont regroupées en lots. Cette sous-section n'a pas pour objectif d'énumérer exhaustivement ces travaux, mais plutôt de pointer les principales approches de modélisation et de résolution. La version classique du problème de regroupement des demandes en lots peut être énoncée comme suit [36] : étant donné un ensemble de demandes à collecter avec, pour chaque demande, son poids et l'ensemble d'items la constituant et leurs emplacements de stockage respectifs et considérant une règle de routage fixée qui permet de définir la tournée d'un préparateur pour récolter les différents items d'un lot de demandes et une capacité limite à ne pas dépasser pour chaque lot, le problème consiste à trouver l'ensemble des lots, et les demandes composant chaque lot, en respectant la contrainte de capacité pour chaque lot et en minimisant la longueur parcourue pour les collecter.

Gademann et Velde [37] ont prouvé que ce problème est NP-difficile au sens fort pour un nombre de demandes par lot supérieur à 2. Ils ont par ailleurs proposé un algorithme *branch and price* pour traiter le problème, avec une stratégie de routage arbitraire. Leur méthode résout des instances comportant jusqu'à 32 clients à l'optimalité. Dû à la complexité du problème, peu de travaux se sont intéressés à une approche exacte pour le résoudre. On peut citer Öncan [38] et Bozer et Kile [39] qui ont proposé des formulations en PLNE avec des stratégies de routage spécifiques. La majorité des articles de la littérature se sont tournés vers des méthodes approchées. On retrouve globalement trois types d'heuristiques constructives : les heuristiques de priorité [40], les heuristiques "seed" [41] et les heuristiques de saving [42]. Les heuristiques de priorité trient les demandes selon une règle de priorité et les affectent séquentiellement aux lots. Les heuristiques "seed" construisent les lots progressivement et travaillent en deux phases : pour chaque lot, une demande "seed" est choisie selon une règle

de sélection. Ensuite, une règle d'accompagnement permet de choisir les autres demandes dans le lot jusqu'à ce qu'aucune demande ne puisse être ajoutée à ce lot sans violer la capacité maximale. Le processus continue avec les autres lots jusqu'à ce que toutes les demandes soient affectées aux lots. Les heuristiques de gains se basent sur l'heuristique de Clarke & Wright [42]. Dans l'étape initiale de l'heuristique, un lot est généré pour chaque demande. Ensuite, à chaque étape, des gains en distance résultant de la fusion de deux lots sont calculés et la fusion qui donne le meilleur gain en distance est appliquée. Le processus s'arrête lorsqu'il n'y a aucune fusion qui puisse être faite sans violer la contrainte de capacité du lot. Par ailleurs, plusieurs métaheuristiques ont été développées pour ce problème. Albareda et al. [43] proposent une heuristique de recherche à voisinage variable qui réduit de 6% en moyenne la solution retournée par l'heuristique de Clarke & Wright sur des instances allant jusqu'à 250 demandes. Henn et al. [44] proposent une heuristique de recherche locale itérée et une heuristique de colonie de fourmis qui produisent de meilleures solutions que l'heuristique de Clarke & Wright combinée à une recherche locale simple. Henn et Wäscher [45] proposent deux algorithmes de recherche avec tabou qui surpassent les méthodes proposées dans [44] en termes de qualité de solution et de temps de calcul sur des instances à 100 demandes. Pour une description plus détaillée des différentes heuristiques et métaheuristiques proposées dans la littérature, on réfère le lecteur intéressé à une revue de la littérature sur ce problème [46].

Les travaux précédents imposent une stratégie de routage simple pour construire la tournée associée à chaque lot. Cependant, le regroupement des demandes en lots et le routage des préparateurs sont des problèmes étroitement liés. Il est donc évident que la considération d'une approche intégrée pourrait potentiellement améliorer la qualité des solutions. Des articles récents s'intéressent de plus en plus à cette problématique intégrée qu'on peut énoncer comme suit : partant des mêmes hypothèses que celles du problème de regroupement des demandes en lots, excepté la présence d'une règle de routage, le problème consiste encore à trouver l'ensemble des lots, et les demandes composant chaque lot, mais en plus à déterminer la tournée de chaque lot avec l'objectif de minimiser la distance totale parcourue.

Kulak et al. [47] ont modélisé le problème à l'aide d'une formulation basée sur le TSP et le *bin packing* pour une disposition à un ou deux blocs. Ils ont par ailleurs proposé un algorithme de recherche avec tabou en incluant deux heuristiques de TSP pour construire les tournées de chaque lot. Valle et al. [48] ont proposé un algorithme de type *branch and cut* renforcé avec des inégalités valides capables de trouver des solutions optimales pour des instances jusqu'à 30 demandes. Les résultats expérimentaux ont démontré que les inégalités valides introduites améliorent considérablement les performances de la méthode. Valle et Beasley

[49] ont proposé une formulation en PLNE dans laquelle la partie routing est approximée. La tournée optimale pour récupérer chaque lot de commandes est ensuite construite dans une deuxième étape. La formulation donne des résultats très compétitifs avec des temps d'exécution faibles. Scholz et Wäscher [50] ont investigué la question du potentiel bénéfice qu'on pouvait obtenir en résolvant le problème de regroupement de demandes en lots et de routage d'une manière intégrée. À travers un algorithme de recherche locale itérée, ils ont comparé la solution retournée en résolvant le problème de routage à chaque itération d'une manière optimale et celle qui résulte d'une application d'une stratégie de routage simple. Les résultats démontrent qu'un gain moyen de 25% sur la distance totale parcourue pouvait être obtenu sans effort de calcul supplémentaire (c.-à-d. pour un même temps de calcul).

Tous les travaux cités précédemment partent de l'hypothèse que les demandes ne peuvent pas être affectées à plusieurs lots (autrement dit, que les items d'une commande puissent être collectés par différentes tournées). Très peu d'articles considèrent cette opportunité. Parmi les quelques articles la considérant, on note l'article de [51] qui étudie le problème de regroupement de demandes en lots et de routage dans un entrepôt d'une société appelée HappyChic en France. Dans leur définition du problème, des boîtes de capacité limitée sont utilisées pour récupérer les commandes. Dû au fait que les commandes sont généralement composées de plusieurs items, plusieurs boîtes peuvent être nécessaires pour prendre en charge une commande. Le problème est donc de trouver une affectation d'items aux boîtes et puis de regrouper les boîtes en lots avec pour objectif de minimiser la distance totale parcourue. Il est à noter que pour des raisons de sécurité, une stratégie de routage fixe est imposée par la société pour le routage d'un préparateur. Le problème est résolu avec une heuristique capable d'améliorer la solution courante implémentée dans l'entrepôt de 24.21 %. Briant et al. [52] ont étudié le problème de regroupement de demandes en lots dans deux industries dont l'une est HappyChic. Ils ont proposé une heuristique de génération de colonnes où le sous-problème correspond à un problème de routage d'un préparateur. Les résultats expérimentaux démontrent qu'une amélioration allant jusqu'à 23.5% pouvait être obtenue avec l'heuristique de génération de colonnes par rapport à la méthode introduite par [51].

Le problème de regroupement des demandes en lots dans les entrepôts à étagères mixtes est relativement nouveau dans la littérature. À notre connaissance, [53] est le premier article qui le considère. Dans la définition de leur problème, un ensemble de demandes doit être récupéré par un préparateur de commandes. Plusieurs dépôts alternatifs sont présents dans l'entrepôt où un préparateur peut déposer une ou plusieurs demandes complètes. La contrainte de capacité dans ce cas est exprimée en termes du nombre de demandes actives. Une demande

est active si elle est en cours de traitement, c'est-à-dire qu'elle a été commencée mais par encore terminée. Si la capacité maximale est atteinte, le préparateur ne peut prendre en charge une nouvelle commande que s'il complète la collecte des items d'une des commandes courantes et la rapporte à un dépôt. Pour résoudre le problème, ils proposent une méthode heuristique qui travaille en deux niveaux : dans le niveau supérieur, un pool de solutions partielles prometteuses est géré. Le pool démarre de solutions vides qui sont étendues de manière progressive. L'extension d'une solution partielle se fait dans le niveau inférieur à l'aide d'une heuristique qui sélectionne un sous-ensemble de demandes qui ne sont pas encore prises en charge par la solution courante, utilise les règles de sélection introduites par [34] pour choisir d'où récupérer les items de chaque demande active et fait appel à une version modifiée du programme dynamique de Ratliff et Rosenthal [27] pour définir la plus courte tournée pour visiter les emplacements sélectionnés (la modification permet à une tournée de finir dans un dépôt alternatif si la distance parcourue est réduite). À chaque itération au niveau supérieur, un nombre fixe de solutions partielles sont sélectionnées aléatoirement et étendues. Ensuite, un mécanisme de notation permet de trier les différentes solutions partielles (anciennes et nouvellement générées) et d'en garder un nombre limité parmi les plus prometteuses. Yang et al. [54] ont étudié la variante à étagères mixtes du problème classique de regroupement des demandes en lots. Ils proposent un ensemble d'heuristiques qui diffèrent dans la règle de sélection des emplacements de stockage pour récupérer chaque item d'un lot. Dans leurs expérimentations, ils démontrent qu'une règle se distingue des autres, à savoir celle qui consiste à : 1. trier les items du lot par ordre croissant du nombre d'emplacements de stockage où ils se trouvent ; 2. construire un tour en prenant les items dans l'ordre établi et en sélectionnant à chaque itération l'emplacement de stockage qui engendre la plus petite contribution à la distance totale de parcours. Cependant, ils considèrent que le nombre de copies d'un item stockées dans un emplacement de stockage est suffisant pour répondre à toute la demande, ce qui n'est généralement pas le cas en pratique.

2.3.3 Problème de regroupement des demandes en lots et de séquençage des lots sur les préparateurs

Lorsque les demandes arrivent avec des dates limites de préparation, l'affectation des lots aux préparateurs de commandes et le séquençage des lots sur chaque préparateur doivent être considérés en plus du regroupement des demandes en lots. Il existe deux manières de prendre en considération cette nouvelle donne en fonction du respect, ou non, des dates limites de préparation des commandes.

Dans la première, on considère que la date limite associée à chaque demande peut être violée et donc l'objectif du problème revient à minimiser la somme des retards. Ce problème, introduit en 1993 par [55], peut être énoncé comme suit : étant donné un ensemble de demandes qui possèdent les mêmes caractéristiques que le problème de regroupement des demandes en lots et de routage, avec en plus une date limite de préparation associée à chaque demande. Étant donné un ensemble de préparateurs de commandes avec des chariots à capacité limitée. Le problème consiste à trouver une partition des demandes en lots, à affecter chaque lot à un préparateur, à séquencer les lots pour chaque préparateur et à déterminer les tournées de chaque lot de manière à minimiser la somme des retards des préparations des commandes. La majorité des approches considèrent, pour la partie de routage, une stratégie fixe de routage simple (et donc ne considèrent pas vraiment le problème de routage intégré). Elsayed et al. [55] étudient le problème avec un seul préparateur de commandes avec pour objectif de minimiser la somme des avances et des retards des commandes (l'affectation des lots aux préparateurs de commandes n'intervient donc pas). Ils proposent une règle de type "seed" pour construire les lots de la solution initiale et une descente locale en utilisant un voisinage de type 'swap" pour améliorer cette solution. Henn et Schmid [56] proposent un algorithme de recherche locale itérée pour le problème avec un seul préparateur de commandes qui réduit de 45% la somme des retards par rapport à [55]. Henn [57] considère une variante du problème avec plusieurs préparateurs et propose un algorithme de recherche à voisinage variable (VNS) et un algorithme de descente locale à voisinage variable (VND) pour la résoudre. Dans la partie expérimentale, il démontre que les deux méthodes améliorent considérablement la solution retournée par une heuristique basée sur des règles de priorité. Parmi les travaux qui considèrent le routage d'une manière intégrée, Chen et al. [8] proposent un algorithme génétique combiné avec un algorithme de colonie de fourmis pour résoudre le problème. Cependant, un seul préparateur de commandes est considéré dans leur étude et les instances sur lesquelles ils ont résolu leur problème sont relativement petites (8 demandes au maximum). Scholz et al. [58] reprennent la VND proposée dans [57] et introduisent deux nouvelles améliorations. La première est l'introduction d'un nouvel opérateur de voisinage qui détruit complètement un lot de demandes et les affecte à différents lots existants. Cet opérateur permet une réduction qui peut aller jusqu'à 63% de la somme des retards. La deuxième est une post-optimisation des tournées associées à chaque lot en utilisant l'heuristique de LKH (voir [28]) à chaque solution localement optimale. Cela permet une réduction massive du retard total allant jusqu'à 95%. Enfin, [59] est la seule référence qui étudie ce problème en autorisant l'affectation des items d'une commande à plusieurs tournées. Cependant, un seul préparateur de commandes est considéré dans la définition de leur problème.

Dans la deuxième approche, les dates limites de préparation sont considérées comme des contraintes dures à respecter. En effet, pour des entreprises comme Amazon, qui font payer les services de livraison rapide à leur clientèle, assurer une qualité de service optimale est primordial. À notre connaissance, seul un article dans la littérature considère les contraintes de date limite comme des contraintes dures. Il s'agit de [60] qui étudie le problème pour une entreprise de e-commerce spécialisée en pièces détachées. Les auteurs proposent un algorithme de recherche locale itérée pour résoudre le problème avec l'objectif de minimiser le temps total de collecte. Les résultats expérimentaux démontrent qu'un gain de 16.9% en moyenne peut être obtenu en utilisant leur méthode par rapport à la méthode courante implémentée dans l'entreprise.

Il est à noter qu'une troisième catégorie de problèmes dans laquelle un ordonnancement des lots sur les préparateurs doit être fait en plus de la construction des lots existe. Ces problèmes n'associent pas une date limite de préparation à chaque commande, mais considèrent plutôt une vague de commandes qui doit être finie le plus rapidement possible. Les vagues sont généralement prédéterminées à un niveau supérieur de telle sorte que les commandes composant chaque vague doivent être livrées ensemble. Ainsi, l'objectif du problème est de minimiser la date de fin de collecte de la vague de commandes. Ardjmand et al. [61] étudient un cas pratique d'un entrepôt où ce problème émerge en proposant un algorithme parallèle de recuit simulé combiné à un algorithme de colonie de fourmis. Ils démontrent que leur méthode améliore de 7.8% la solution courante de l'entreprise et estiment que cette réduction génère des gains de l'ordre d'une centaine de milliers de dollars annuellement. Rasmi et al. [62] étudient le même problème dans les entrepôts à étagères mixtes avec deux objectifs conflictuels, à savoir la minimisation de la date de fin de la collecte et la minimisation du nombre de préparateurs. La partie expérimentale s'est principalement axée sur le potentiel bénéfice d'adopter une stratégie de stockage à étagères mixtes par rapport à des stratégies classiques sur des instances caractéristiques du pattern de demande des détaillants du e-commerce (peu d'items par commande). Les résultats démontrent qu'adopter une stratégie de stockage à étagères mixtes donne de bien meilleures performances. Cependant, il est difficile de juger à quel point ses conclusions sont pertinentes, car la problématique définie est complexe et la qualité de la solution proposée n'a pas été validée (surtout pour la version du problème qui ne considère pas une stratégie de stockage à étagères mixtes).

2.4 Conclusion et positionnement des travaux par rapport à la littérature

Nous avons constaté à travers la revue de la littérature que l'entreposage en général et le processus de préparation des commandes en particulier ont beaucoup été étudiés par la communauté scientifique. La majorité des travaux se sont cependant concentrés sur des problématiques émergentes dans des entrepôts classiques, à savoir des entrepôts dans lesquels un nombre *très limité* de *grosses* commandes doivent être collectées.

Avec la globalisation, l'émergence de l'e-commerce et des services de livraison rapide, un nouveau type de demande a vu le jour à travers lequel un grand nombre de petites demandes doivent être récupérées, chacune venant avec une date limite de préparation à respecter. Pour le moment, un ensemble de pratiques implémentées dans les industries d'e-commerce ont peu été prises en compte dans la littérature scientifique.

À travers cette thèse, nous allons d'abord étudier, dans le chapitre 4, le potentiel bénéfice d'affecter les items d'une commande à plusieurs tournées. Pour cela, nous allons considérer un entrepôt qui reçoit un ensemble de demandes avec chacune une date limite de préparation à ne pas dépasser. Le problème étudié dans ce chapitre est un problème de regroupement d'items de demandes en lots, de séquençage des lots sur les préparateurs et de routage des préparateurs avec pour objectif la minimisation du temps total parcouru. Il s'agit d'une généralisation du problème introduit dans [60] dans lequel chaque demande doit être prise en charge dans son intégralité par un préparateur de commandes et dans une seule tournée. Pour résoudre le problème étudié, nous proposons une heuristique en deux étapes : dans la première étape, un ensemble de tournées est construit en adaptant la procédure de découpage de tour géant couramment utilisée pour résoudre des problèmes de tournées de véhicules [63]. Ensuite, les tournées construites sont ordonnancées sur les préparateurs de commandes en utilisant une formulation de programmation par contraintes. Dans la partie expérimentale, nous comparons la solution produite par notre heuristique avec celle produite par l'algorithme de recherche locale itérée de [60]. Les résultats ont démontré qu'une réduction de 30% en moyenne du temps total de collecte pouvait être observée, avec une réduction maximale allant jusqu'à 60%.

Dans le chapitre 5, nous proposons une méthode de décomposition de Benders de type "logic-based" pour la résolution du problème de routage d'un préparateur de commandes dans les entrepôts adoptant une politique de stockage à étagères mixtes. À notre connaissance, un seul article a proposé une méthode exacte pour résoudre ce problème dans la

littérature [35]. Notre méthode, à la différence de la formulation en MILP de [35] qui travaille exclusivement sur des entrepôts arrangés en un bloc, prend en charge des entrepôts à plusieurs blocs. Dans ce chapitre, le problème étudié consiste à collecter un ensemble d'items, parfois en plusieurs exemplaires. De plus, chaque item est présent en quantité limitée dans plusieurs emplacements de stockage dans l'entrepôt. L'objectif est donc de sélectionner les emplacements d'où récupérer les items et de déterminer la tournée pour visiter les différents emplacements sélectionnés. Dans la méthode proposée, les décisions sur la sélection des emplacements de stockage pour satisfaire la demande sont prises en charge dans le problème maître tandis que le sous-problème construit la tournée associée à la solution retournée par le problème maître. Vu qu'aucune procédure générique ne permet de produire des coupes de Benders dans des méthodes de type "logic-based", nous proposons une coupe adaptée à notre problème dont nous prouvons la validité. De plus, nous proposons un ensemble de techniques d'amélioration incluant une fonction de génération de bornes inférieures et des inégalités valides et nous verrons dans la partie expérimentale que ces techniques sont la clé de la performance de notre méthode. De plus, nous adaptons la formulation en MILP de [10] au contexte des entrepôts à étagères mixtes que nous utiliserons comme base de comparaison avec notre méthode.

Dans le chapitre 6, nous étudions la combinaison de deux pratiques utilisées dans les entrepôts de e-commerce, à savoir le découpage de l'entrepôt en zones et le stockage à étagères mixtes. On a pu voir à travers la littérature que c'est une combinaison qui n'a jamais été étudiée et que la récente revue de [22] a identifiée comme une intéressante perspective de recherche. Dans la définition de notre problème, on se positionne dans un environnement dynamique où des vagues de demandes urgentes doivent être récupérées à chaque nouvelle tournée d'un ensemble de préparateurs. De plus, l'entrepôt est partitionné en zones et un préparateur est affecté à chaque zone. L'objectif du problème est alors de minimiser la date de fin de collecte de l'ensemble des demandes d'une vague qui correspond à la tournée la plus longue des préparateurs. Pour résoudre le problème, on propose d'adapter la méthode développée dans le chapitre 5 pour le problème du routage d'un seul préparateur. De plus, on proposera une autre méthode spécialisée pour les entrepôts en un bloc qui correspond à une adaptation de la méthode introduite dans [35] pour le routage d'un préparateur. Dans la partie expérimentale, on s'intéresse à comparer la performance du processus de collecte dans un entrepôt implémentant une stratégie de stockage traditionnelle avec un entrepôt à étagères mixtes.

CHAPITRE 3 DÉMARCHE ET ORGANISATION DE LA THÈSE

Comme le présente la revue de la littérature, le processus de préparation des commandes a reçu un grand intérêt dans la littérature scientifique. Cependant, la majorité des travaux ont pour but de proposer des modèles et des méthodes permettant d'organiser ce processus dans des entrepôts traditionnels. Comme mentionné dans l'introduction, le commerce électronique donne naissance à un nouveau pattern de demandes poussant les entrepôts à repenser leurs techniques de stockage et de préparation de commandes. Cette thèse de doctorat s'est intéressée à trois problématiques, dont deux ont donné lieu à des articles soumis dans des journaux scientifiques. Ces travaux de recherche ont pour but de proposer des modèles et des méthodes intégrant un ensemble de techniques implémentées par des entrepôts modernes et d'étudier leur pertinence.

Le chapitre 4 présente le premier projet de cette thèse. La motivation majeure de ce projet, et de cette thèse en général, est de poser des hypothèses de travail à la fois novatrices d'un point de vue académique et pertinentes d'un point de vue pratique. Après une étude et une analyse approfondie de la littérature scientifique, nous avons identifié une pratique émergente dans les entrepôts d'e-commerce mais encore peu étudiée, à savoir la décomposition des items d'une commande sur plusieurs préparateurs. Les travaux de recherche présentés dans ce chapitre comportent deux objectifs majeurs. Le premier étant de proposer une modélisation d'un problème de planification de la préparation des commandes associées au e-commerce où chaque commande est traitée en se donnant la possibilité de répartir ses items sur plusieurs préparateurs. Le deuxième étant d'étudier l'impact de cette pratique sur les performances du processus global de préparation des commandes. Pour répondre au premier objectif, nous avons supposé connaître l'ensemble de demandes arrivant avec des dates limites de préparation et l'ensemble de préparateurs munis de chariots de capacité limitée pour définir le problème de regroupement d'items en lots, d'ordonnancement des lots sur les préparateurs et de routage des préparateurs. Pour résoudre ce problème complexe, nous nous sommes naturellement tournés vers des méthodes dites heuristiques pour apporter des solutions de bonne qualité pour des instances représentant des entrepôts réels. À cet effet, nous avons proposé une heuristique composée de deux étapes, la première étape construit des tournées de collecte et la seconde affecte ces tournées aux différents préparateurs. Cette heuristique exploite plusieurs techniques et méthodes connues en recherche opérationnelle dont certaines ont été adaptées et améliorées. Pour répondre au deuxième objectif, nous avons identifié dans la littérature une étude qui s'intéresse à une version de ce problème qui n'autorise pas

la décomposition des items d'une commande et pour laquelle un grand ensemble d'instances, représentant un entrepôt d'e-commerce vendant des pièces détachées, est disponible. Nous avons donc mené une étude comparative entre la solution de notre méthode et celle proposée dans cet article sur les instances disponibles. Les résultats ont démontré qu'un gain considérable en termes de temps de collecte (30 % en moyenne et pouvant aller jusqu'à 60%) était obtenu en autorisant la répartition des items d'une commande sur plusieurs préparateurs au lieu de l'interdire, ce qui démontre la pertinence de cette pratique.

Le chapitre 5 présente le deuxième projet de cette thèse. Dans ce projet, nous avons voulu concevoir un modèle et des méthodes pour optimiser le processus de préparation de commandes en prenant en compte une nouvelle technique de stockage, à savoir le stockage à étagères mixtes. Nous avons initialement prévu d'incorporer cette technique dans la définition du premier problème étudié précédemment, mais nous avons constaté que le problème devenait extrêmement complexe, rendant difficile la tâche de concevoir une méthode de résolution rapide et efficace, ce qui est un impératif d'un point de vue managérial pour un problème de planification de ce type. Pour palier à cette difficulté, nous avons opté pour une approche optimisant tournée par tournée et nous nous sommes donc intéressés à la résolution d'un problème de conception de tournée d'un préparateur à partir d'une liste d'items à collecter. L'objectif de ce problème est de sélectionner les emplacements à visiter pour récupérer chaque item et de concevoir la tournée qui démarre d'un dépôt, visite les emplacements sélectionnés et retourne à ce même dépôt. Ce problème a déjà été formulé dans la littérature scientifique (PRP-MS). Cependant, à notre connaissance, la seule méthode exacte résolvant ce problème opère sur un type particulier d'entrepôt, à savoir des entrepôts répartis en un seul bloc. Dans ce travail, nous avons proposé deux méthodes exactes capables de résoudre le problème pour des entrepôts plus génériques avec un nombre arbitraire de blocs. Dans la première méthode, nous avons exploité le fait que le problème est décomposable en un problème maître (sélection des emplacements de collecte) et un sous-problème (conception de la tournée de collecte) pour proposer une méthode de décomposition de Benders de type "logic-based" (LBBD). Nous avons rapidement vu qu'une implémentation directe de cette méthode conduisait à des solutions de très mauvaise qualité. Le défi majeur dans ce projet a été de proposer des techniques d'accélération qui permettraient de rendre la méthode efficace. Ce défi a été relevé en introduisant une fonction de génération de bornes inférieures et des inégalités valides. Dans la deuxième méthode, nous avons proposé une adaptation d'un PLNE de l'état de l'art, initialement conçu pour résoudre le PRP classique, à ce problème de collecte dans des entrepôts à étagères mixtes. Dans la partie expérimentale, les deux méthodes ont été comparées sur des instances de la littérature et des instances nouvellement

générées. Les résultats démontrent que la première méthode (LBBD) obtient clairement de meilleurs résultats que la deuxième.

Le chapitre 6 présente quant à lui le dernier projet de cette thèse. L'objectif de ce projet est de mesurer le gain qui pourrait être obtenu en passant d'un stockage traditionnel à un stockage à étagères mixtes pour le processus de préparation de commandes. Pour cela, nous avons généralisé le modèle introduit dans le chapitre précédent en considérant un ensemble de préparateurs devant récupérer une vague de commandes dans leur prochaine tournée. L'idée est de considérer que les commandes arrivent en temps réel et que la planification du processus de préparation se déroule par vagues successives de commandes. À chaque vague de commandes, nous déterminons une tournée par préparateur pour récupérer l'ensemble des items des commandes de la vague. L'objectif du problème est de minimiser le temps de collecte pour récupérer tous les items de la vague de commandes, qui est équivalent à minimiser la longueur de la plus longue tournée parmi les préparateurs. Lors de cette étude, nous avons également considéré une autre technique d'organisation permettant de réduire la congestion dans les allées et étant peu considérée dans la littérature, à savoir le découpage de l'entrepôt en zones et l'affectation d'un préparateur à chaque zone. Pour résoudre le problème, nous avons adapté deux méthodes existantes, initialement conçues pour résoudre le PRP-MS. La première est une adaptation de la méthode de décomposition proposée dans le chapitre précédent et la deuxième est une adaptation d'un PLNE de la littérature spécialisé pour les entrepôts constitués d'un seul bloc. Les deux méthodes ont été validées sur un ensemble d'instances nouvellement générées. De plus, les expérimentations ont démontré que le passage d'un stockage traditionnel à un stockage à étagères mixtes permettrait d'enregistrer une réduction considérable du temps de collecte.

Les principaux résultats et limitations des différents travaux sont discutés dans le chapitre 7. Enfin, les conclusions de la thèse sont présentées dans le chapitre 8.

CHAPITRE 4 ARTICLE 1 : THE INTEGRATED ORDERLINE BATCHING, BATCH SCHEDULING, AND PICKER ROUTING PROBLEM WITH MULTIPLE PICKERS: THE BENEFITS OF SPLITTING CUSTOMER ORDERS

M. Haouassi, Y. Kergosien, J. Mendoza et L-M. Rouseau ont écrit cet article et l'ont publié le 1 juillet 2021 dans *Flexible Services and Manufacturing Journal*

4.1 Introduction

Warehousing involves four main activities: receiving, storing, picking, and shipping goods [1]. Among all these activities, order picking is unanimously considered as the most time-consuming and costly process. It can induce up to 65% of all labor activities [64] and 70% of the total operating costs [47]. Two classes of systems can be distinguished to process the picking activities in the warehouses: manual picking systems and automated picking systems [36]. In this paper, we focus on the manual picking system, more precisely a picker-to-part system, which is the most common in practice [13]. It is characterized by (human) order pickers starting from the depot, walking or driving a trolley through aisles to pick stock-keeping units (SKUs) from storage locations, and coming back to the depot. Pickers are directed using picking lists where the orderlines to retrieve in a single tour and their respective storage locations are itemized. A picking list can be composed of orderlines of a single order (pick by order) or orderlines of multiple orders (pick by batch).

In the last decades, a new retail model known as "e-commerce" has grown significantly throughout the world. Canadian e-commerce retailers for instance sold for almost 40 billion $ of goods in 2018, accounting for 8.1% of the Canadian total retail sales. It represents a 9.1% increase over the previous year (36.6 billion $ in 2017), and it is expected to reach 56.6 billion $ in 2023. Similar growth rates can be observed on a more global scale, where business to costumer e-commerce sales have generated almost 995 billion in 2015, and are expected to exceed 1.5 trillion $ in 2018 [65].

The demand pattern in e-commerce warehouses is characterized by many but small, time critical customer orders to process [66]. Shifting toward e-commerce requires combining a wide range of planning problems in integrated approaches. The recent survey of Van Gils et al. [9] classifies the main tactical and operational planning problems that occur in e-commerce

warehousing and shows the correlation between these problems. In this study, we focus on the integration of 3 planning problems: regrouping the customer orders into pick lists (order batching), designing the tour that retrieves each pick list (picker routing), and assigning the picking lists to a set of pickers and scheduling the lists assigned to each picker (picker scheduling) in order to optimize a performance criterion.

A very large part of the picking literature assumes the integrity of the customer orders when performing the picking process, *i.e.*, the orderlines of customer order have to be retrieved in a single tour. It is generally argued by the fact that splitting the customer orders over several batches will increase human errors or cause unacceptable packing efforts [56,58,67]. Nonetheless, new sorting technologies and logistical techniques have emerged in the e-commerce warehouses making the order accumulation and packing procedure more efficient. For instance, some warehouses use fully automated conveyors that sort the orderlines according to the belt [68]. This mechanism, although it is efficient, requires high investment costs. Other warehouses of business-to-customer (B2C) segment (Amazon Europe, Zalando) apply a manual consolidation and packing process based on the so-called "put walls" [2]. Figure 4.1 schematizes this process. A logistical worker, named "putter" receives orderlines from the picking area and stores each orderline in a dedicated shelf of the put wall. On the other side of the put wall, another logistical worker named "packer" retrieves each completed order, packs it, and sends it to the shipping area. A put-to-light mechanism is used to guide the putter and packer in their tasks. This system is particularly relevant in the e-commerce context since the characteristics of the orders (small sizes and tight shipping times) avoid the overload of the put wall.

We aim through this work to demonstrate the benefit of splitting the customer orders when organizing the picking process in e-commerce warehouses. We thus generalize the integrated order batching, batching routing, and picker scheduling problem of Van Gils et al. [60] by allowing the splitting of customer orders and propose a route first-schedule second (RFSS) heuristic to solve the problem. In the computational experiments, we show that reductions up to 60% of the total order picking time are obtained by splitting the customer orders using our heuristic compared to the ILS of Van Gils et al. [60] for the non-splitting version of the problem.

The remainder of this paper is organized as follows: in Section 4.2, we formally describe the orderline batching, batch routing, and picker scheduling problem. A MILP formulation of the problem is given in appendix A.1. Section 4.3 comprises a literature review regarding the

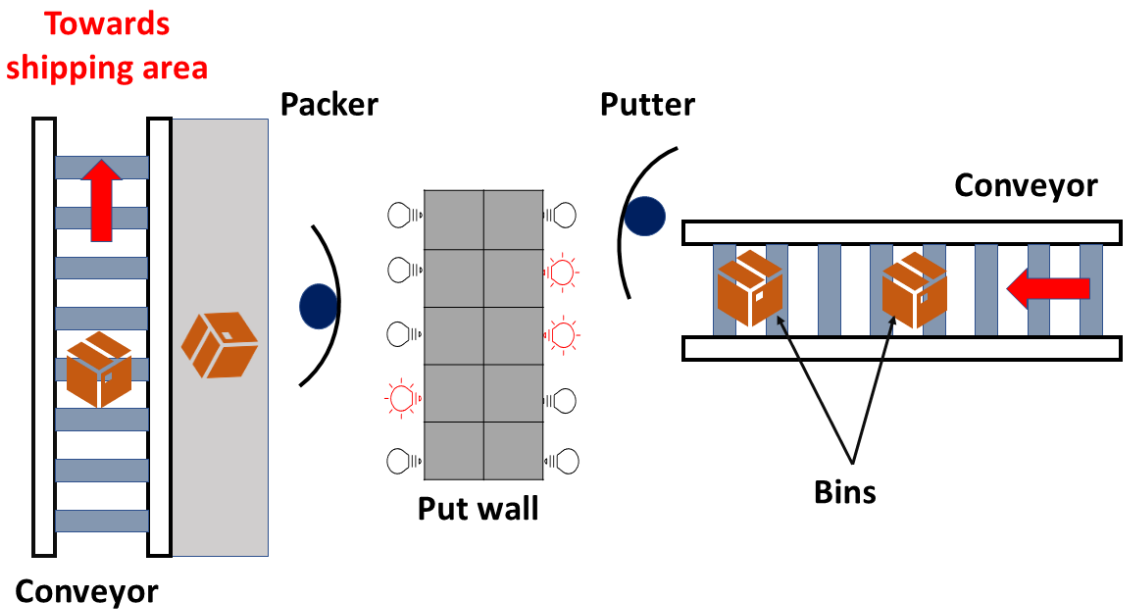


Figure 4.1 Consolidation and packing schema using put walls [2]

traditional and e-commerce warehouses. Section 5.3 presents a route first-schedule second heuristic to solve the problem. In Section 5.6, the numerical results of our heuristic are presented and compared with the results of [60] to demonstrate the benefits of splitting the customer orders. Finally, section 6.6 provides conclusions and outlines research perspectives.

4.2 Problem Statement

We tackle an order picking problem in a warehouse of e-commerce segment characterized by:

— **A low-level picker-to-part system:** in which order pickers start from a central **depot**, travel through **picking** and **cross-aisles** to pick SKUs, and return to the depot. The SKUs are stored in shelves (storage locations) directly accessible to the pickers without using a fork truck. The cross-aisles divide the picking area into blocks and the part of a picking aisle located between two cross-aisles is named **subaisle**. Figure 4.2 sketches an example of a warehouse with a 2-block layout. The "pick locations" represent the stop-points from where the pickers retrieve the SKUs. A set of storage locations is associated with each pick location (red arrow in the Figure 4.2). We assume that the aisles are large enough to ignore the effect of pickers blocking.

— **Time-critical picking orders:** Online retailers promise to customers short time frames between the order "click" and the "knock" on the door announcing its delivery. For instance, Amazon offers to its customers to deliver the orders requested online in the next day or even in the same day in some regions. Satisfying this promise is a main factor for the brand management of the retailer. We thus assume that deadlines are associated with the orders. The deadline of each order is assumed to be fixed in an upper stream level according to the departure time of the vehicle that delivers this order.

— **Small orders:** the orders in e-commerce segment consist of few orderlines. For instance, the average number of orderlines in the Amazon warehouse in Germany is 1.6 SKUs [34].

— **Mixed-shelves storage policy:** In e-commerce warehouses, a common and efficient practice is to break up unit-load of an SKU into small quantities that are scattered through multiple shelves in the warehouse. This storage policy is referred to as mixed-shelves storage policy. It increases the probability to find SKUs of a pick list close by irrespective of the position in the warehouse [66]. However, the picking location of each orderline has to be selected when assuming this storage policy. To keep the problem tractable, we assume in our problem definition that the picking location of each orderline is selected in an upperstream level (*i.e.*, a fixed picking location is associated with each orderline such as storage constraints are satisfied).

Formally, we consider a complete and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that models the warehouse layout, where $\mathcal{V} = \{0\} \cup \mathcal{L}$ is the node set and $\mathcal{E} = \{(i,j) \in \mathcal{E} : i,j \in \mathcal{V}, i < j\}$ the edge set. In $\mathcal{V}$, node 0 represents the depot while the set $\mathcal{L}$ represents the pick locations where the pickers stop to collect the SKUs around them. A non-negative weight $t_{i,j}$ is associated with each edge (i,j), and it represents the travel time between nodes i and j. We assume that the pickers travel at a constant speed, and hence the travel times are proportional to the travel distances.

Let $\mathcal{O}$ be the set of customer orders. Each order $o \in \mathcal{O}$ consists of few orderlines $\mathcal{M}_o$ that must be retrieved before a deadline d_o. An orderline m corresponds to a particular SKU and has two associated parameters: a weight q_m and a dedicated pick location $l_m \in \mathcal{L}$ from where to retrieve it.

The customer orders are processed by a set of homogeneous order pickers $\mathcal{K}$ equipped with carts of capacity W. Each picker performs picking tours in the warehouse starting and ending

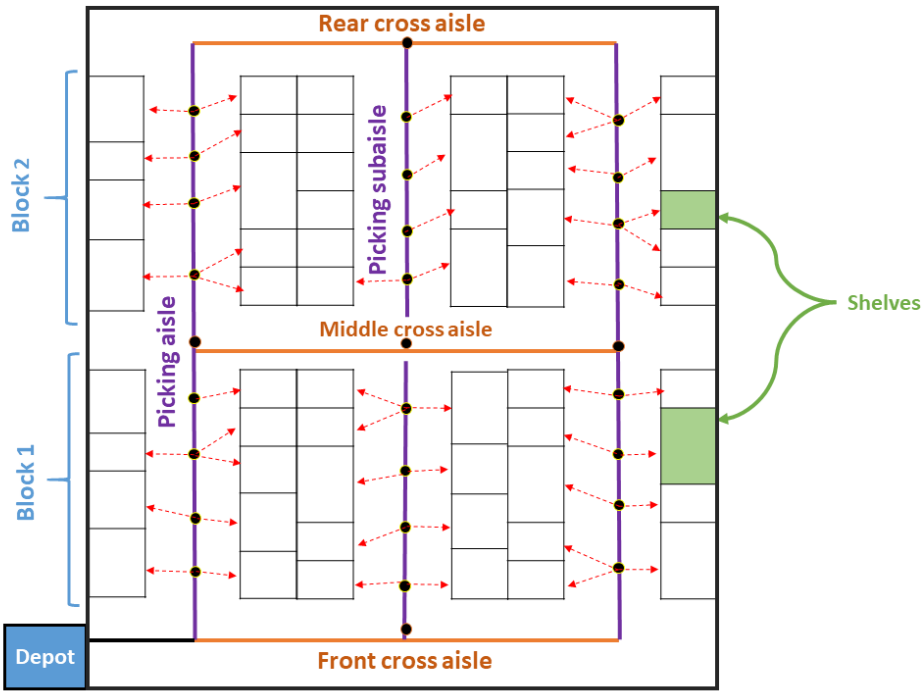


Figure 4.2 Two blocks warehouse layout

at the depot. For each tour v, a set of orderlines $\mathcal{M}_v$ is retrieved from their corresponding pick locations in a predefined sequence. The processing time p_v of a tour v is defined as the sum of:

— a fixed setup time β^s to prepare the tour;

— a fixed search and pick time β^p for each picked orderline;

— the travel time to walk between depot and the first pick location plus the travel time between the pick locations according to the sequence visit plus the travel time between the last pick location and depot.

We aim at constructing a picking plan $\Pi = \{\Pi_k\}_{\{k \in \mathcal{K}\}}$, where Π_k corresponds to a set of sequenced picking tours that must be performed by the picker k during his/her work-shift. Π must satisfy the following constraints:

1. Each order is completely picked before its deadline. Note that since the splitting of a customer order is allowed, the completion time of an order o equals the completion time of the last tour that processes an orderline in $\mathcal{M}_o$
2. The charge of each tour v, which is equal to the cumulative weight of its orderlines $\mathcal{M}_v$, does not exceed the cart capacity

3. The tours assigned to each picker do not overlap. Let consider v and v' two consecutive tours assigned to the picker k, with et_v the completion time of v and $st_{v'}$ the start time of v', then: $st_{v'} \geq et_v$
4. Each tour starts and ends at the depot

Finally, Π is an optimal planning schedule of our problem if it minimizes the total processing time (order picking time) function Z.

$$Z(\Pi) = \sum_{k \in \mathcal{K}} \sum_{v \in \Pi_k} p_v \tag{4.1}$$

4.3 Literature Review

In traditional warehouses, a small number of customer orders are processed daily and each one is made of a multiple orderlines and high quantities [69]. To optimize the picking operations in this context, the picking literature focuses on two main problems depending on the nature of the picking lists: When assuming a pick by order system, the arising optimization problem is named the picker routing problem and can be stated as follows: given a set of storage locations to visit in a pick list, what sequence of visits minimizes the total processing time. This problem is modeled in the literature as a TSP problem in a complete and symmetric graph in which the nodes represent depot and storage locations. The arc weights of the graph are fixed by computing the shortest travel time between each pair of nodes. The picker routing problem is also modeled as a Steiner-TSP problem in a rectangular warehouse (see Figure 4 of [50]). A collection of exact and heuristic methods exists to solve this problem. Ratliff and Rosenthal [27] introduce a fundamental theorem that limits the number of possible configurations for traversing an aisle in an optimal solution to 6 configurations. Furthermore, they propose a dynamic programming algorithm (DP) that solves the problem for a particular warehouse layout called one block layout warehouse. Their work served as a basis for several extensions in the picker routing literature: In [11], an extension to warehouses with two blocks layout is proposed; In [70], the algorithm is adapted to support an arbitrary starting and ending point; In [71], the problem based on warehouses with two blocks, parallel but non equal-sized subaisles is solved; Finally, in [72], the DP of Ratliff and Rosenthal [27] is extended to the so-called chevron warehouses which were proposed by Öztürkoğlu et al. [73] as a design option for unit-load warehouses with single-command operations. Pansart et al. [29] propose two exact methods to solve the picker routing problem: a Steiner-TSP formulation reinforced with warehouse-based valid inequalities and an adaptation of a dynamic programming algorithm for solving the rectilinear-TSP inspired from [31]. They showed that the dynamic programming algorithm outperforms (in CPU time) the efficient TSP solver

"CONCORDE" for instances with less than 11 blocks. In practice, the solutions returned by exact formulations can be rejected by managers since they produce "illogical" picking tours and hence difficult to memorize [50]. The pickers are more familiar with the so-called "routing strategies". Routing strategies are warehouse-based heuristics that model intuitive human practices such as: traversing entirely any aisle (except the last one) that contains at least one orderline to pick (S-shape strategy); entering and exiting each aisle that contains an orderline to pick from the same point (return strategy)... A comparative study between those methods and the efficient LKH (Lin-Kernighan-Helsgaun) heuristic for TSP problems can be found in [32]. The picker routing problem was extended by Löffler et al. [74] for AGV-assisted order picking systems where a set of customer orders has to be picked in a single tour. The orders have to be sequenced and all the pick locations of an order o must be visited before processing the next order $o + 1$ of the sequence. They propose an adaptation of the DP of Ratliff and Rosenthal [27] to solve the special case when orders sequence is assumed to be fixed and imbed it within a greedy heuristic to solve the general problem.

When the customer orders are medium-sized and the pickers are equipped with relatively high-capacity carts, a pick by batch system in which orders are regrouped into batches is set up to decrease the processing times. The order batching problem (OBP) seeks to regroup a given set of orders into batches that satisfy a maximum capacity and construct the tour associated to each batch in order to minimize the total processing time. This problem is proven strongly NP-hard for more than two orders by batch [37]. The picking literature proposes a rich body of exact and heuristic methods to solve the OBP. The constructive heuristics can be classified into 3 categories: "priority heuristics" in which the orders are sorted according to a sorting rule and then assigned greedily to empty batches according to an assignment rule; "seed heuristics" in which each batch is first initialized with seed order, then filled according to order congruency rule; and "saving heuristics" in which the Clarke & Wright heuristic for the VRP (see [42]) is adapted to the OBP. Henn et al. [46] give a detailed review of the several heuristics and metaheuristics developed for the OBP. Most of the existing work assumes a fixed routing strategy for computing the processing time of each batch. To study the potential reduction of total processing time that can be achieved by implementing more sophisticated routing methods, Scholz and Wäscher [50] propose an iterated local search algorithm embedded with several heuristics to compute the tours. They conclude that average savings of 25% can be obtained when the tours are computed using near-optimal method.

In e-commerce warehouses, the 3 planning problems presented in the introduction section (order batching, picker routing and picker scheduling) have to be considered in an integrated fashion. In [57], due dates are associated with the customer order and the total tardiness of customer orders is minimized. To solve the problem, two metaheuristics based on variable neighborhood descent (VND) and Variable neighborhood search (VNS) starting with the same initial solution are proposed. The neighborhood structures embedded in the methods are classified in two groups: the first group changes the position of a complete batch or swaps two batches in the current solution and the second group moves a customer order to another batch or swaps two customer orders of two distinct batches. Experiments show that even with higher computational efforts, the VNS is outperformed by the VND. Furthermore, the neighborhood structures improve significantly (40% on average) the initial solution generated by a priority-rule based heuristic. In [58], the same problem is solved with a metaheuristic based on VND. Their work can be outlined in three main contributions. First, a new and efficient heuristic able to reduce the total processing time of the initial solution of Henn [57] by up to 63%. Second, a new neighborhood that breaks up a complete batch and assigns its orders to other batches is introduced and represents the largest proportion in the total tardiness reduction. Finally, the tour that processes each batch is recomputed in each local optimum solution using the LKH-heuristic leading to a massive reduction of the total tardiness (up to 95%). From a managerial point of view, the customers satisfaction is the main performance factor in e-commerce retail. Managers in that field prefer to increase the labor insensitivity of the picking process (number of order pickers) instead of accepting solutions that may violate the shipping deadlines of some orders. Given this observation, Van Gils et al. [60] introduce the integrated batching, routing and scheduling problem with hard time constraints and total processing time minimization. An Iterative Local Search heuristic (ILS) is developed to solve the problem and applied to a spare part e-commerce warehouse. Experiment results report savings of 16.9% on average by using the ILS algorithm compared to the priority-based rule usually applied in the warehouse.

A limited number of articles in the literature allows the splitting of customer orders in the picking process. Among these exceptions, Bué et al. [51] (resp. Briant et al. [52]) proposes a heuristic (resp. exact formulation) for solving a variant of the order batching problem in a men's ready-to-wear warehouse named HappyChic. In their problem definition, capacitated boxes are used to pick the customer orders. Since the orders are large, they generally cannot be regrouped in a single box. The problem is thus to find which customer orders to split into a limited number of boxes and regroup the boxes into batches in order to minimize the total processing time. Note that due to safety considerations related to the warehouse layout, the

processing time of each batch is computed using a fixed routing strategy. Furthermore, some managerial restrictions impose to consider a minimum volume constraint on boxes. Tsai et al. [59] study an integrated order batching, batch routing, and picker scheduling problem with orders splitting. However, since they assume a single order picker, the assignment of batches to pickers is not considered. Their objective function to minimize is defined as a linear combination of the total processing time and the orders earliness and tardiness. To solve the problem, they propose a multiple-Genetic-Algorithm. Their computational experiments were devoted to study the impact of simultaneously considering the distinct terms of the objective function instead of considering them one by one. To the best of our knowledge, no approach in the literature studies the impact of splitting the customer order in e-commerce warehouses where a set of small and time critical customer orders has to be picked by a set of order pickers operating simultaneously.

4.4 Route first-Schedule second heuristic

We propose in this section a matheuristic to tackle the problem. Algorithm 4.1 describes the general structure of our approach. It is composed of two phases: routing phase (lines 1-5) and scheduling phase (line 6). The routing phase regroups the customer orders into clusters and constructs a set of picking tours for each cluster of orders. The clusters are built using the procedure `constructClusters`($\mathcal{O}$, γ). This procedure assembles greedily the orders that have close deadlines. To build the picking tours associated with each cluster, we ignore the time constraints to obtain a variant of the order batching problem in which we allow the orders splitting when we construct the picking tours (batches). This problem is solved using the procedure `orderlinesBatching-SplitBasedProcedure`($\mathcal{O}^c$). The latter starts by computing a giant tour that starts from depot, retrieves all orderlines of the current cluster of orders, and comes back to depot. It then uses an adaptation of the split procedure by [75] to optimally extract a set Π^c of tours that satisfy the capacity constraints from the giant tour. It is worth noting that our method differs from the classical version of the split algorithm in the computation of the arcs' costs in the auxiliary graph as will be explained in Subsection 4.4.1. The tours generated for each cluster are added to the solution Π. After finding the tours of all clusters, the routing phase ends and the scheduling phase starts. In this phase, we model the picking tours in Π as jobs and the pickers in $\mathcal{K}$ as machines. Each job has a processing time (the processing time of its associated tour) and a deadline (the shortest deadline of the orders processed in its associated tour). We thus assign the picking tours in Π to the order pickers and schedule the tours assigned to each order picker solving a constraint programming model. Note that since the processing times of the jobs (*i.e.*,, the

constructed tours) are fixed during the routing phase, the scheduling problem does not have an objective function (it is a decision problem). In the remainder of this section, we describe the main algorithmic components.

Algorithm 4.1 Matheuristic: the general structure

Require: $\mathcal{G}, \mathcal{O}, \mathcal{K}, \gamma$

1: $\mathcal{C} \leftarrow$ `constructClusters`$(\mathcal{O},\ \gamma)$
2: **for** $\mathcal{O}^c \in \mathcal{C}$ **do**
3: $\quad \Pi^c \leftarrow$ `orderlinesBatching-splitProcedure`$(\mathcal{O}^c)$
4: $\quad \Pi \leftarrow \Pi \cup \Pi^c$
5: $s^* \leftarrow$ `toursScheduling` $(\Pi, \mathcal{K})$
6: **return** s^*

4.4.1 Routing phase

Order clustering

Algorithm 4.2 gives the pseudo-code of the procedure `constructClusters`$(\mathcal{O},\ \gamma)$. The procedure is based on a maximum capacity of a cluster $W^{cluster} = \gamma \cdot K \cdot W$, with $\gamma \geq 1$ being a parameter of the algorithm. First, the customer orders in $\mathcal{O}$ are sorted in increasing order of their deadlines. Next, the clusters are constructed greedily one at a time. Starting from an empty cluster, the procedure adds orders to the cluster while there is slack on the capacity constraint. If adding an order o violates the maximum cluster capacity, we close the current cluster and open a new empty cluster. The process is repeated until all customer orders are assigned to a cluster. Note that the value of parameter γ impacts the feasibility and the quality of the final solution. Indeed, setting γ to a large value leads to a small set of large clusters (*i.e.*, clusters with a large number of orderlines). From such set of clusters, the solution to the orderline batching problems would probably contain better tours in terms of the objective function but may lead to an infeasible final solution. Indeed, since this resolution ignores the deadlines of the customer orders, the orderlines that have close deadlines may be spread across too many tours, compared to the number of pickers. Therefore, it may be impossible to schedule their picking before the deadline.

Orderline Batching

Let $\mathcal{M}^c$ be the set of orderlines of the customer orders in cluster c. The orderline batching problem seeks to regroup $\mathcal{M}^c$ into a set Π^c of batches (tours) that satisfies the cart capacity and minimizes the total processing time. To perform this task, the procedure

Algorithm 4.2 `constructClusters`($\mathcal{O}$, γ)

$\mathcal{C} = \emptyset$, $w = 0$, $c = 0$, $\mathcal{O}^c = \emptyset$
sort $\mathcal{O}$ according to the earliest deadline rule.
for $o \leftarrow 1$ *to* $\mathcal{O}$ **do**
    while $(w \leq \gamma \cdot K \cdot W)$ **do**
        $\mathcal{O}^c \leftarrow \mathcal{O}^c \cup \{o\}$
        $w \leftarrow w + w_o$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{O}^c$
    $c \leftarrow c + 1$
    $\mathcal{O}^c = \emptyset$
    $w \leftarrow 0$
return $\mathcal{C}$

`orderlinesBatching-SplitBasedProcedure`($\mathcal{O}^c$) starts by computing a giant tour that retrieves all the orderlines in $\mathcal{M}^c$ using the LKH, an efficient heuristic for the TSP. According to a comprehensive computational experiment conducted by [32], LKH produces near-optimal solutions for the picker routing problem (*i.e.*„ solutions with an average 0.1% deviation with respect to the optimal solution).

LKH returns a giant tour to pick all orderlines in $\mathcal{M}^c$, we denote m_i^c the orderline index at the i^{th} position in the giant tour ($\forall i \in \{1, \ldots, |\mathcal{M}^c|\}$ $m_i^c \in \mathcal{M}^c$). The procedure `orderlinesBatching-SplitBasedProcedure`($\mathcal{O}^c$) then aims to solve a specific shortest path problem in an auxiliary direct and acyclic graph $\mathcal{G}^c = \{\mathcal{V}^c, \mathcal{A}^c\}$ where $\mathcal{V}^c = \{m_0^c, m_1^c, \ldots, m_{|\mathcal{M}^c|}^c\}$ is the node set and $\mathcal{A}^c$ the arc set. The node $m_0^c \in \mathcal{V}^c$ is a dummy node while $\{m_1^c, m_2^c, \ldots, m_{|\mathcal{M}^c|}^c\}$ represents the orderlines of the giant tour. The arc set $\mathcal{A}^c$ is defined as $\mathcal{A}^c = \{(m_i^c, m_{i'}^c)\ \forall i, i' \in \mathcal{M}^c / i < i'\}$ and as each arc $(m_i^c, m_{i'}^c) \in \mathcal{A}^c$ models a feasible picking tour $v_{(i,i')}$ that starts from the depot, retrieves the orderlines $(m_{i+1}^c, m_{i+2}^c, \ldots, m_{i'}^c)$ and comes back to the depot. A cost $\tilde{p}_{(m_i^c, m_{i'}^c)}$ is associated with each arc $(m_i^c, m_{i'}^c)$, and it represents the processing time of the tour $v_{(m_i^c, m_{i'}^c)}$. The classic version of the split algorithm computes the travel time of the tour $v_{(m_i^c, m_{i'}^c)}$ by assuming that the nodes between m_{i+1}^c and $m_{i'}^c$ are visited according to their associated sequence in the graph (direct arc computation). To improve this version, we adopt a different approach. As per the observation of [32], combining a warehouse-based routing strategy and the best improvement 2-opt operator leads to high-quality solutions and negligible computational times for the picker routing problem. We decided to reevaluate the travel time of each arc by using a combined heuristic + 2-opt operator procedure. Combined heuristic is a hybrid routing strategy that combines S-shape routing strategy and return routing strategy. An algorithmic description of the combined heuristic can be found in [76]. Note that we observed in preliminary tests that the travel

time of a tour $v_{(m_i^c,m_{i'}^c)}$ returned by the direct arc computation may outperform the reevaluated travel time in very rare cases. Therefore, the procedure that computes the travel time of each tour $v_{(m_i^c,m_{i'}^c)}$ (noted `Compute-Travel-Time`$(m_i^c, m_{i'}^c)$) selects the best value between the direct arc computation version and the reevaluated version.

Algorithm 4.3 `orderlinesBatching-SplitBasedProcedure`$(\mathcal{O}^c)$

$\mathcal{V}^c \leftarrow$ compute giant tour using the LKH heuristic
$\tilde{t}_0 = 0$
for $i \leftarrow 1$ *to* $|\mathcal{V}^c|$ **do**
    $\tilde{t}_{m_i^c} = \infty$
    $P_{m_i^c} = -1$
for $(i \leftarrow 0$ *to* $|\mathcal{V}^c| - 1)$ **do**
    $i' \leftarrow i + 1$
    $load \leftarrow 0$
    while $(i' \leq |\mathcal{V}^c|$ **and** $load + w_{m_{i'}^c} \leq W)$ **do**
        $load = load + w_{m_{i'}^c}$
        $\tilde{p}_{(m_i^c,m_{i'}^c)} \leftarrow \beta^s + \beta^p \cdot (i' - i)+$ `Compute-Travel-Time`$(\mathcal{G}^c, m_i^c, m_{i'}^c)$
        if $\tilde{t}_{m_i^c} + \tilde{p}_{(m_i^c,m_{i'}^c)} < \tilde{t}_{i'}$ **then**
            $\tilde{t}_{m_{i'}^c} = \tilde{t}_{m_i^c} + \tilde{p}_{(m_i^c,m_{i'}^c)}$
            $P_{m_{i'}^c} = m_i^c$
        $i' = i' + 1$
Construct Π^c from the labels
return Π^c

To find the optimal splitting of the giant tour into several feasible tours, the split procedure aims to find the shortest path from m_0^c to $m_{|\mathcal{M}^c|}^c$ in $\mathcal{G}^c$. Algorithm 4.3 describes the whole procedure based on a label setting algorithm. A label $(\tilde{t}_{m_i^c}, P_{m_i^c})$ is associated to each node m_i^c, $\tilde{t}_{m_i^c}$ is the value of a path from m_0^c to m_i^c and $P_{m_i^c}$ is the predecessor node from which $\tilde{t}_{m_i^c}$ has been determined. After computing the giant tour, the labels are initialized. Then two nested loops update the labels. The main loop traverses the nodes from m_0^c to $m_{|\mathcal{M}^c|-1}^c$. At each iteration i of the main loop, the inner loop explores all the arcs (the routes) that share the same tail node m_i^c and satisfy the cart capacity. For each feasible arc $(m_i^c, m_{i'}^c)$, $\tilde{p}_{(m_i^c,m_{i'}^c)}$ is computed and the label of the head node is updated using the bellman optimality principle. The picking tours in Π^c are finally constructed by a backward pass through the predecessor labels and starting by the label $(\tilde{t}_{m_{|\mathcal{M}^c|}^c}, P_{m_{|\mathcal{M}^c|}^c})$.

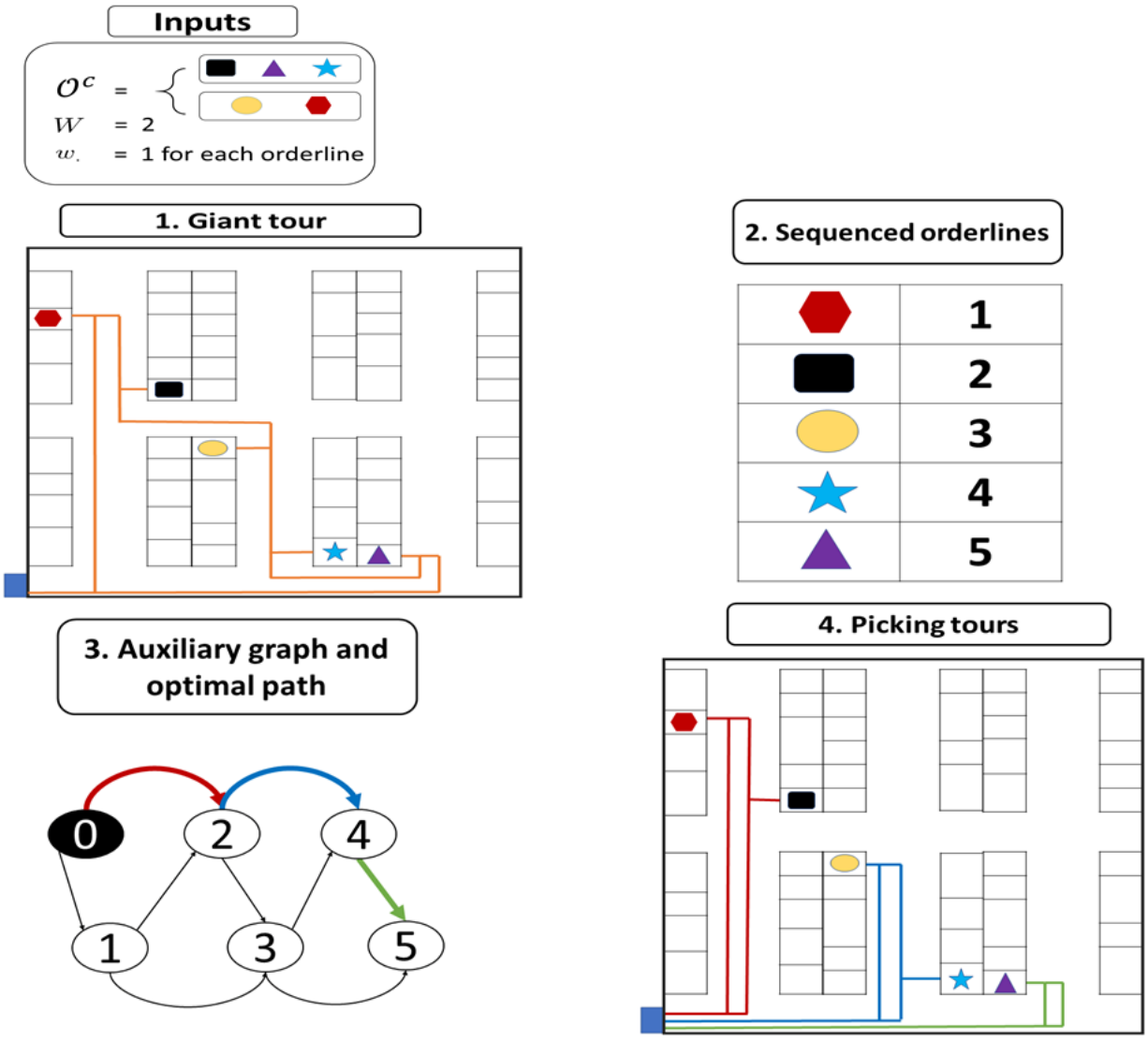


Figure 4.3 Split-based procedure: graphical example

Figure (4.3) depicts an example of the splitting procedure for a cluster of orders composed of 2 orders, the first one consists of 3 orderlines and the second one consists of 2 orderlines. The cart capacity is 2 orderlines. The path represented by the orange lines shows the optimal TSP-like tour to retrieve all the orderlines. Furthermore, the red, blue, and green paths show the optimal picking tours associated with the red, blue, and green arcs of the optimal path in the auxiliary graph.

4.4.2 Scheduling phase

The second phase problem is a variant of the identical parallel machine scheduling problem with deadlines and without objective function (decision problem). Each tour $v \in \Pi$ represents a job to schedule on one of the K machines (the pickers). To tackle the problem, we adopt a constraint programming (CP) approach. Constraint programming has been applied successfully to a variety of scheduling problems such as: manufacturing, computer and network scheduling, transportation,etc. [77–80].

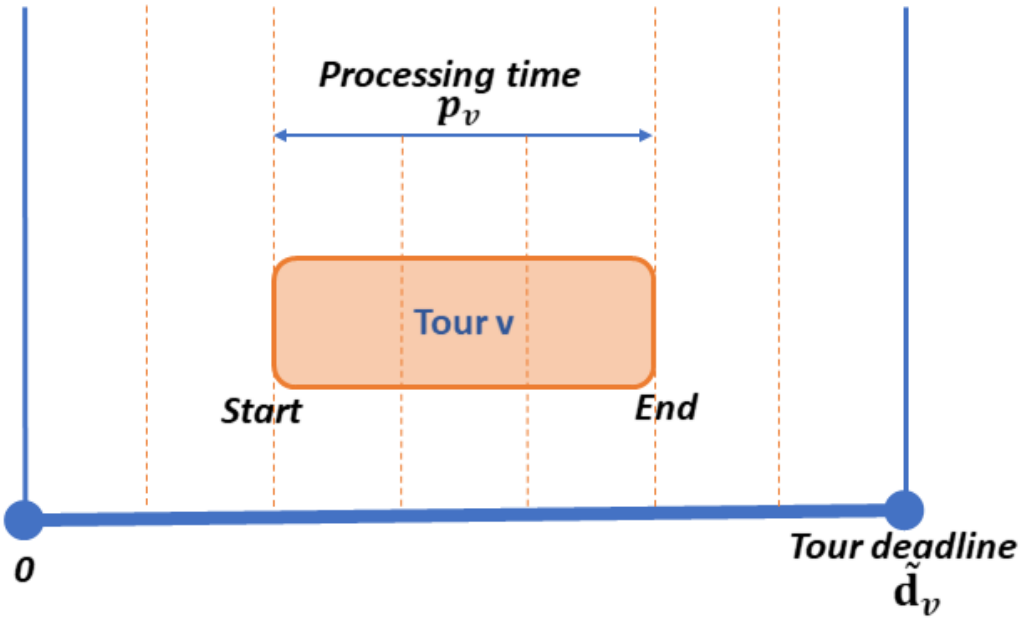


Figure 4.4 Representation of interval variable x_v

For the sake of brevity we do not present our CP formulation here, but the interested reader can find it in Appendix A.3. To implement and solve our model, we exploit the scheduling capabilities of CP Optimizer (CPO). In particular, we use interval variables, sequence variables, and global constraints. The CP model re-written using CPO objects reads:

Table 4.1 Notations used in the CP formulation

Parameters	
$\tilde{d}_v$	Deadline of a tour v ($\tilde{d}_v = \min_{o \in \mathcal{O}_v} d_o$)
Variables	
$y_{k,v}$	Optional interval variable when tour v is assigned to picker k with duration p_v and domain $\{[0, p_v), [1, p_v + 1), \ldots, [\tilde{d}_v - p_v, \tilde{d}_v)\}$
x_v	Interval variable associated with the tour v (not optional)
$Z_k = \{y_{k,1}, \ldots, y_{k,\lvert\Pi\rvert}\}$	Set of optional interval variables (sequence variable) that models the sequenced tours of picker k

$$\texttt{Alternative}(x_v, \{y_{1,v}, \ldots, y_{K,v}\}) \qquad \forall v \in \Pi \tag{4.2}$$

$$\texttt{NoOverlap}(Z_k) \qquad \forall k \in \mathcal{K} \tag{4.3}$$

Interval variables are a powerful tool for modelling scheduling problems. They embed several attributes of a job such as start time, end time, and processing time in a single decision variable [81]. Figure (4.4) depicts a representation of an interval variable x_v for a picking tour v. The processing time of tour v defines the size of its associated interval variable. The time interval (start time, completion time) of tour v is fixed by assigning values to the attributes "start" and "end" of the interval variable. Since the value "end" is bounded by the tour deadline, the domain of an interval variable associated with tour v is defined by the discrete set $\{[0, p_v), [1, p_v + 1) \ldots, [\tilde{d}_v - p_v, \tilde{d}_v)\}$. One feature of an interval variable is that it can be optional. In this case, an empty decision is added to its domain to model the case where the interval variable is absent in the schedule. We use this feature to model the assignment of picking tour v to picker k (variables $y_{k,v}$). We also use sequence variable Z_k to order the interval variables assigned to each picker k.

Constraints (4.2) are alternative constraints. They force each tour v to be assigned to exactly one picker. It works as follows: for a given v, if x_v is present (which is always the case since x_v is not optional) then exactly one of the elements of $\{y_{1,v}, \ldots, y_{K,v}\}$ will be present in the final schedule. Constraints (4.3) prevent the overlapping of interval variables in each sequence variable $\mathbf{Z_k}$. If the problem is unfeasible then no solution is returned.

4.5 Computational experiments

The computational experiments aim to show the benefits of splitting the customer orders when building the picking routes. We thus test our heuristic (RFSS) over the data set used in [60] and compare our results with the results of their ILS heuristic. The instances are described in Section 4.5.1. Then, the benefits of recomputing the arc costs of the split graph is showed in Section 6.5.2. Next, parameter tuning and sensitivity analysis are discussed in Section 4.5.2. Finally, the benefits of splitting the customer orders is shown in Section 4.5.4.

All computations were performed on a 64-bit laptop equipped with an intel core i7-8550U CPU (1.80 GHz), 16 gigabytes main memory, running on Windows 10. Our RFSS heuristic is coded in C++ (visual studio 2019), using IBM ILOG CP Optimizer 12.8 to solve the constraint programming problems. After preliminary experiments, we fixed a time limit of 30 s for the constraint programming solver and assume that our algorithm returns infeasible solution if the constraint programming solver reaches the time limit without finding a feasible schedule of the picking tours. Finally, note that the computations in [60] were performed on an Intel Xeon Processor $E5-2680$ (2.8 GHz) using a single thread.

4.5.1 Test problem instances

The data set generated by [60] is referred to as `Set-VG`. The data represent a 2-block warehouse layout. Storage locations are homogeneous and assumed to be 1.3 m long and 0.9 m wide. Each picking aisle is 3 m wide. The distance between two adjacent picking aisles is thus set to 4.8 m. The depot is located in the front of the leftmost aisle and the distance between the depot and the closest picking point is 15.65 m (12 $m + 3.65$ m). The two blocks are separated by 6 m. Figure 4.5 sketches out all these metrics. Each picker travels at a constant speed of 1 $m \cdot s^{-1}$ and needs 10 s to identify and pick each orderline. Furthermore, 180 s are required to prepare each tour.

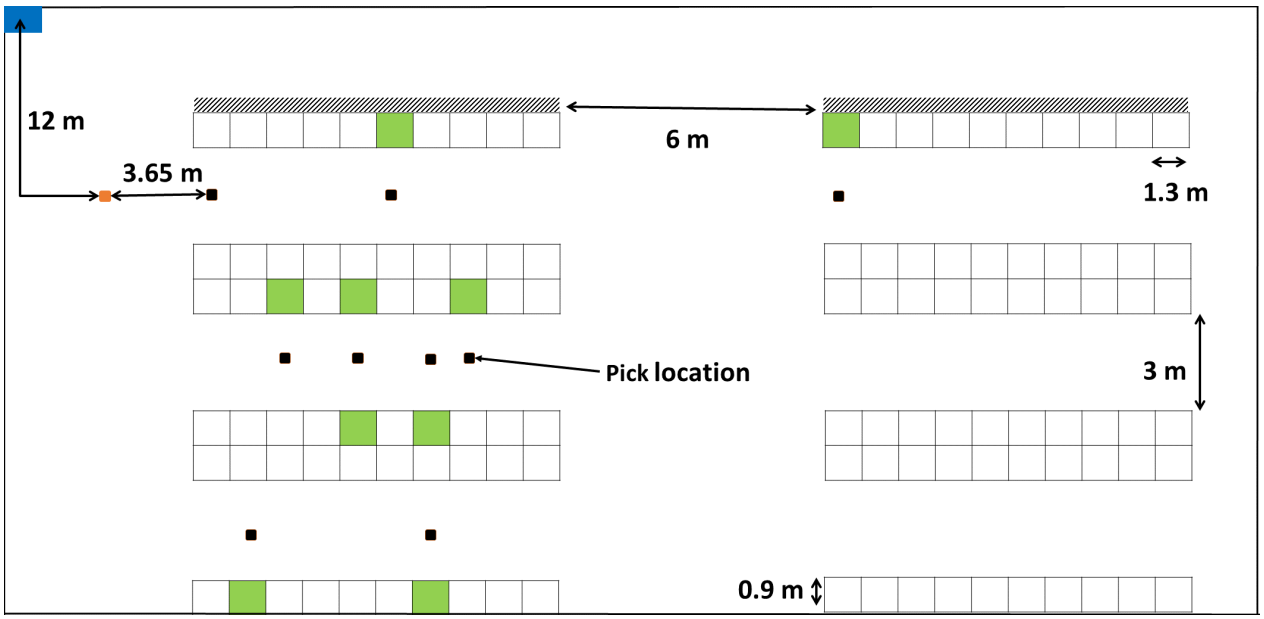


Figure 4.5 Warehouse floor in `Set-VG`

In order to cover multiple warehouses scenarios, six factors (warehouse layout, storage policy, cart capacity, order structure, deadline distribution) have been varied during the instances generation process, with three levels for each factor. For the warehouse layout, the number of subaisles and storage locations per subaisle is varied resulting in small warehouses (SW) with 12 subaisles and 60 storage locations in each subaisle, medium warehouses (MW) with 24 subaisles and 120 storage locations in each subaisle, and large warehouses (LW) with 36 subaisles and 180 storage locations in each subaisle. The assignment of SKUs to storage locations is done using random (Rand), within-aisle (WA), and across-aisle storage (AA) policies. WA and AA policies are class-based storage policies where the picking area is divided into 3 classes (A,B,C). Class A includes $\frac{1}{6}$ of all SKUs with the highest order frequency (60%) whereas classes B and C include $\frac{1}{3}$ and $\frac{1}{2}$ of SKUs with order frequencies of 30% and 10% respectively. All the orderlines are assumed to have the same weight. The cart capacity is limited to 15, 30, or 45 orderlines, depending on the instance. Similarly, the number of orders is fixed to 100, 200 or 300. For each order o, the number of orderlines $|\mathcal{M}_o|$ is computed using the following formula: $min(W, \lfloor Exp(\beta) + 0.5 \rfloor)$, with $Exp(\beta)$ an exponential distribution with mean $\beta = \frac{8}{3}$ for 300 orders, $\beta = 4$ for 200 orders, and $\beta = 8$ for 100 orders. The planning horizon is set to 4 h and the deadlines are generated using a uniform distribution (Uni), a progressive distribution (Prog), and a degressive distribution (Deg). Using the progressive (resp. degressive) distribution, the highest proportion of orders has to be retrieved at the beginning

(resp. at the end) of the planning horizon. A combination of the above-mentioned parameters results in 243 problem classes. 30 instances are generated for each problem class resulting in 7290 single instances. The number of pickers varies between instances and is fixed using the following regression equation: $K^{(i)} = \lceil 1.20(0.254M + 0.006OL + 0.072W + 1.383Deg) \rceil$ with M the number of aisles, OL the total number of orderlines of instance (i), W the capacity of the cart, and Deg representing the degressive distribution. The coefficients of the equation are obtained using a regression analysis on a set of 30 instances randomly selected from `test-VG`. Note that `test-VG` is a set of 243 generated test instances, each one of them corresponds to a distinct problem class. The number of pickers $K^{(i)}$ of each selected instance (i) is computed as the smallest value of K that enables the ILS heuristic of [60] to return a feasible solution in terms of the the deadline constraints (*i.e.*, for $K^{(i)} - 1$, the ILS heursitc of [60] returns a solution with positive tardiness). Note that the number of pickers of each instance obtained by the regression equation is increased by 20% to ensure a feasible solution for each instance (*i.e.*, enough pickers to pick all orderlines in time). For more details about the data set generation, please refer to [60]. The benchmark is publicly available at `https://www.uhasselt.be/Datasets-and-results`.

4.5.2 Parameter tunning and sensitivity analysis

We study in this section the impact of parameter γ on the solution feasibility, the solution quality, and the computational time of our RFSS heuristic. Recall that since the number of pickers for each instance is fixed, the maximum weight of a cluster of orders is proportional to γ (see section 4.4.1). For instance, assume that $\gamma = 3$. If we assume that the tours constructed from a cluster of orders should be executed in a relatively small time window and the workload is uniformly balanced between the pickers, then each cluster (except the last one) contains enough orders for constructing at least the three next tours of each picker. Firstly, we execute the RFSS heuristic over `Set-VG` data set by setting γ to the following values $(1.5, 1.8, 2.0, 2.5, 2.8, 3.0, 4.0, best)$, where the value *best* corresponds to a parallel multi-start version of our heuristic in which we run our algorithm for each $\gamma \in \{1.5, 1.8, 2.0, 2.5, 2.8, 3.0, 4.0\}$ and store the best feasible solution found. Table 5.2 reports the total number of unsolved instances for each parameter value and Figure 4.6 gives the proportion of best solutions found for each $\gamma \in \{1.5, 1.8, 2.0, 2.5, 2.8, 3.0, 4.0\}$. Second, we remove all the instances for which the algorithm returns an infeasible solution for a given value of γ (1515) and study the average order picking time and CPU time for each parameter value on the remaining instances (5775). The main results are reported in Figure 4.7.

Table 4.2 Number of unsolved instances per parameter value

γ	# of unsolved instances
1.5	0
1.8	0
2.0	0
2.5	1
2.8	4
3.0	56
4.0	1514
best	0

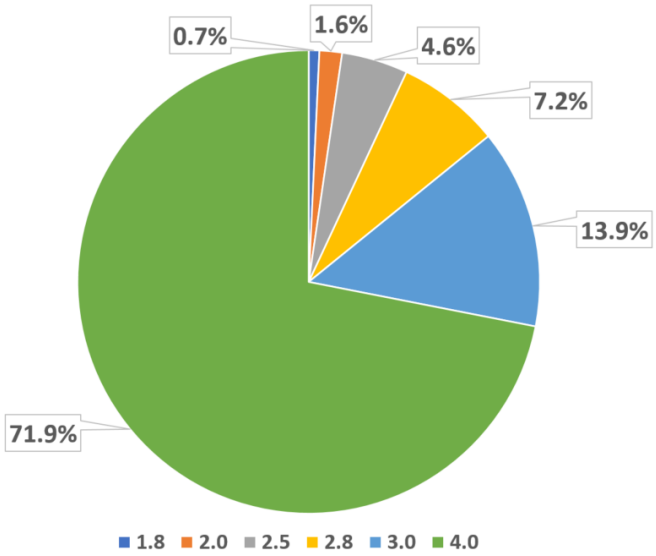


Figure 4.6 Distribution of best solutions per parameter value

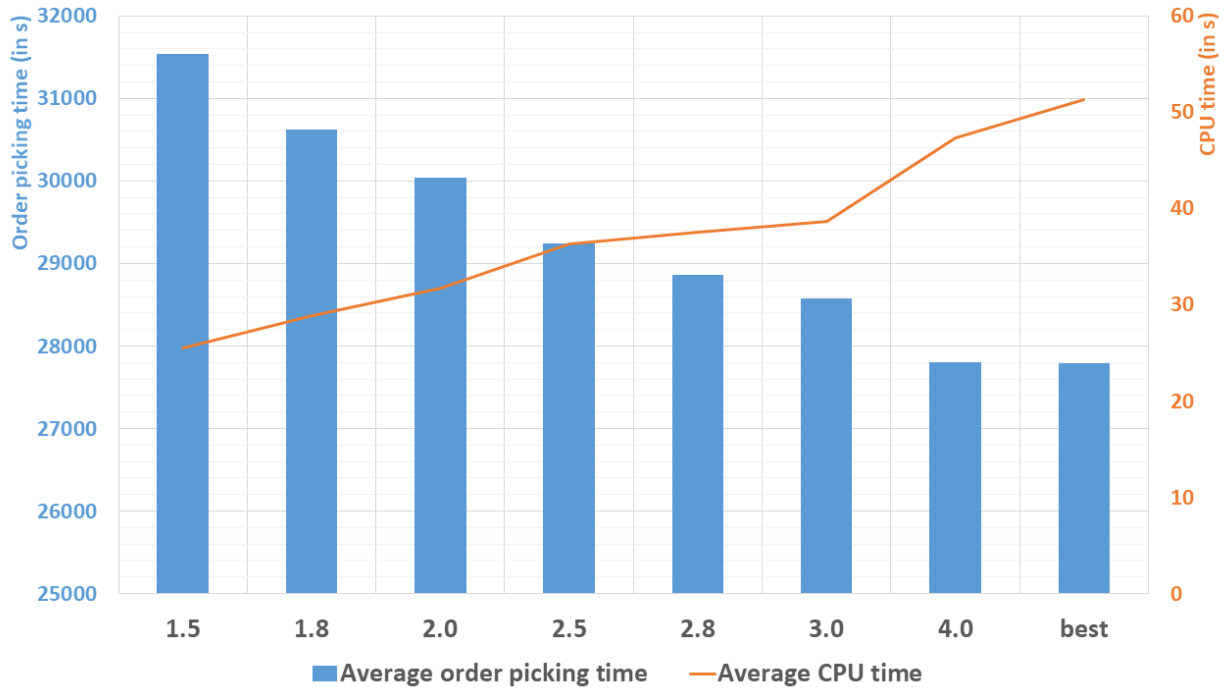


Figure 4.7 Impact of parameter γ on the performance of the RFSS heuristic on the selected set of instances

From Table 5.2, we observe that setting $\gamma \leq 2$ leads to feasible solutions for all the 7290 instances. Infeasible solutions start to appear when $\gamma = 2.5$ and the number of infeasible instances increases exponentially with the increase of γ. Those results were expected, since the batching procedure ignores the order deadlines when generating the picking tours. Thus, short-but-incoherent picking tours are produced when the clusters are too large. It follows that the constraint programming procedure is unable to produce a feasible final solution with those tours. However, we observe that our algorithm was able to find feasible solutions for almost all instances when $2 < \gamma \leq 3$ and for 79.2% of instances even when $\gamma = 4$. Hence, we conclude that even by assembling the orders into few-but-large clusters and ignoring the deadlines during the construction of the picking tours, our algorithm is able to return a feasible (and high-quality) solution for a considerable proportion of instances. This result gives an indication on how splitting the customer orders reduces the effect of the order deadline constraints during the picking process.

Figure 4.7 reveals that increasing γ leads to a reduction in the average order picking time. This result can be explained by the fact that the procedure `constructClusters`($\mathcal{O}$, γ) produces few-but-large clusters when γ is set to high values. Consequently, the search space of the orderline batching problem is increased, leading to the generation of more efficient picking tours. This observation is confirmed by the results in figure 4.6 in which we observe that 97.7% of the best solutions are found when $\gamma > 2$, 71.9% when $\gamma = 4$. Focusing on execution times, we observe a linear increase in the computational time with the increase of γ. This result is not surprising, since the highest portion of the computational time is consumed by the LKH heuristic employed to build the giant tours of each cluster. Larger clusters therefore result more complex TSP problems to solve during this phase of the algorithm.

Given the previous observations, we conclude that setting γ to higher values results in better solutions (on average) but also a higher risk of finding infeasible solutions. To extract a feasible solution with a high quality, we decided to set $\gamma = best$ (parallel multi-start version) which represents the best trade-off between the solution feasibility and the solution quality. Furthermore, we can observe from figure 4.7 that the increase of the average computational time of RFSS when setting $\gamma = best$ is negligible compared to the average computational time of RFSS when $\gamma = 4$. Thus, in the remainder of the manuscript, we use the parallel multi-start version of our algorithm to prove the benefits of splitting the customer orders. We refer to this version of our algorithm as $RFSS^{+}$.

Table 4.3 Comparaison between the performance of $RFSS_c^+$ and $RFSS^+$

		$RFSS_c^+$		$RFSS^+$		$\Delta_{CV\|IV}$	
		cost(s)	time(s)	cost(s)	time(s)	mean	max
Layout							
	SW	21387.8	29.2	21327.5	34.8	0.3%	1.8%
	MW	27303.7	55.4	26986.5	55.8	1.3%	9.0%
	LW	33868.5	82.2	33231.1	77.8	2.0%	13.0%
Storage policy							
	AA	26925.5	44.4	26774.4	85.5	0.6%	11.5%
	Ran	28690.7	38.5	28172.2	43.6	1.8%	13.0%
	WA	26943.8	83.9	26598.6	39.3	1.2%	12.0%
Cart capacity							
	15	36674.0	31.3	36382.8	37.4	0.7%	3.8%
	30	25104.2	55.4	24732.4	58.6	1.3%	9.9%
	45	20781.8	80.1	20429.9	72.4	1.5%	13.0%
Order structure							
	100	24770.0	61.5	24436.4	50.0	1.3%	12.7%
	200	27999.4	56.5	27664.7	56.2	1.2%	13.0%
	300	29790.6	48.8	29444.0	62.2	1.1%	12.5%
Deadline distribution							
	Deg	27254.7	56.9	26952.8	59.8	1.1%	12.7%
	Prog	27532.2	49.6	27202.6	56.7	1.2%	12.0%
	Uni	27773.1	60.2	27389.8	52.0	1.3%	13.0%
All instances		27520.0	55.6	27181.7	56.1	1.2%	13.0%

4.5.3 Classic versus improved split

In this section, we show the impact of recomputing the cost of each arc of the split graph on the feasibility and the quality of the final solution. We thus execute our algorithm using the classic version of the split algorithm and compare it with the improved version. For the sake of brevity, we refer to the version of our algorithm that uses the classic split as $RFSS_c^+$ and keep $RFSS^+$ for the version that uses the improved split. Table 5.3 presents the main results. It reports the average picking time, the average CPU time for each version, and the savings indicators (mean, max.) for each factor level. Note that if we note $cost_{CV}([.])$ the order picking time returned by $RFSS_c^+$ for instance [.] and $cost_{IV}([.])$ the order picking time returned by $RFSS^+$ for the same instance, then the saving $\Delta_{CV|IV}([.])$ is computed with the following formula:

$$\Delta_{CV|IV} = \frac{cost_{CV}([.]) - cost_{IV}([.])}{cost_{CV}([.])} \tag{4.4}$$

Globally, we observe that recomputing the arc costs of the split graph improves the total picking time by 1.2% on average, with a maximal improvement of 13%. Notice that in very few cases (19/7290 instances), the total order picking time returned by $RFSS_c^+$ is better than the one returned by $RFSS^+$. This is due to the fact that for those instances, the solution returned by $RFSS_c^+$ and the one returned by $RFSS^+$ do not correspond to the same value of γ. Indeed, using the improved version of split results in no feasible solutions when setting γ to large value. Thus, the solution returned by the $RFSS^+$ corresponds to a medium or small value of γ. On the other hand, $RFSS_c^+$ returns feasible solutions when setting γ to large value for those instances.

If we analyze the improvement indicators (mean, max.) per factor, we observe that the impact of recomputing the cost of each arc of the split graph is more significant on the instances with a large warehouse layout, for the instances with a random storage assignment policy, and for the instances with a cart capacity of 45 orderlines, which globally represent the conditions of e-commerce warehouses. By restricting the computations for the instances of large warehouses with a random storage assignment policy and a cart capacity of 45 orderlines (270 instances), we found that the mean gap increases to 3.7%. For the order structure and the deadline, we do not observe a statistically significant correlation between the factor levels and the improvement indicators. Concerning the CPU time, we observe that recomputing the arc costs of the split graph does not generate a significant computational overhead. For some instances, we found that $RFSS^+$ outperforms $RFSS_c^+$ in term of CPU time. Indeed, for each of those instances, the constraint programming solver quickly finds a feasible solution for each value of $\gamma \in \{1.5, 1.8, 2.0, 2.5, 2.8, 3.0, 4.0\}$ when running $RFSS^+$ contrary to $RFSS_c^+$ where the constraint programming solver reaches the time limit without finding a feasible solution for some values of γ.

4.5.4 The benefits of splitting the customer orders

To evaluate the potential benefits of splitting the customer orders during the picking process, we compare the solution returned by $RFSS^+$ and the solution returned by the state-of-the-art ILS of [60] for the non-splitting version. Table 4.4 summarizes the main results. The first part of the table includes the average order picking time (cost), the average computational time (time), and the average number of created tours (T) for each method and factor level. Note that the ILS of [60] is referred to simply as ILS. The second part of the table includes the mean savings, the min. savings, and the max. savings obtained by comparing the order picking time returned by ILS and the order picking time returned by

Table 4.4 Comparison between the performance of $RFSS^+$ and ILS

	ILS			$RFSS^+$			$\Delta_{ILS\|RFSS^+}$		
	cost(s)	time(s)	T	cost(s)	time(s)	T	mean	min	max
Layout									
SW	26283.1	118.4	36	21327.5	34.8	36	19%	8%	30%
ML	39348.8	125.4	36	26986.5	55.8	37	32%	19%	50%
LW	53683.0	127.1	37	33231.1	77.8	37	38%	23%	60%
Storage policy									
AA	37901.6	139.5	36	26774.4	85.5	37	28%	13%	51%
Ran	43788.9	117.0	37	28172.2	43.6	37	34%	14%	60%
WA	37624.4	114.3	36	26598.6	39.3	37	27%	8%	50%
Cart capacity									
15	50681.0	69.9	58	36382.8	37.4	58	27%	12%	44%
30	37246.1	127.5	31	24732.4	58.6	31	31%	11%	54%
45	31387.7	173.4	21	20429.9	72.4	21	32%	8%	60%
Order structure									
100	38466.9	55.1	32	24436.4	50.0	33	34%	13%	60%
200	40346.2	113.7	37	27664.7	56.2	37	29%	11%	54%
300	40501.8	202.1	40	29444.0	62.2	40	26%	8%	48%
Deadline distribution									
Deg	39891.5	116.0	37	26952.8	59.8	37	30%	12%	60%
Prog	39471.5	140.6	36	27202.6	56.7	37	29%	8%	57%
Uni	39951.8	114.2	37	27389.8	52.0	37	29%	9%	57%
All instances	39771.6	123.6	36	27181.7	56.1	37	30%	8%	60%

$RFSS^+$. Note that if we denote $cost_{ILS}([.])$ the order picking time returned by ILS for instance [.] and $cost_{RFSS^+}([.])$ the order picking time returned by $RFSS^+$, then the saving $\Delta_{ILS|RFSS^+}([.])$ obtained for this instance is computed with the following formula:

$$\Delta_{ILS|RFSS^+} = \frac{cost_{ILS}([.]) - cost_{RFSS^+}([.])}{cost_{ILS}([.])} \tag{4.5}$$

Additionally, non-parametric tests (test of Kruskal-Wallis, test of Dunn) are performed to analyze the effect of warehouse factors on the order picking time, on the CPU time, and on the gap $\Delta_{ILS|RFSS^+}$. The main results are provided in Appendix A.2.

Reduction of order picking time

We observe from Table 4.4 that splitting the customer orders during the picking process using our algorithm leads to a massive reduction of the order picking time. Indeed, $RFSS^+$

improves the solution without splitting of ILS by 30% (on average) with a minimal and maximal improvement of 8% and 60%. In terms of workload, the order picking time reduction can reach up to 11.5 h by splitting the customer orders and it equals approximately 6 h on average. If we normalize the results by the number of pickers, we find that the picking time of each picker can be reduced by more than 1 h (on average) with a maximum reduction of 1.8 h (for a planning horizon of 4 h). In practice and compared to the scenario where the splitting order is not allowed, even if splitting orders implies an additional time to gather all the items of a same order before finalize the preparation of the order, the gain in the order picking time is large enough to justify considering it in practice.

We observe that the reduction in order picking time is significantly higher for medium and large warehouses compared to small warehouses. It is probably due to the fact that small warehouses contain fewer storage locations leading to a higher probability of finding orderlines of distinct orders stored in the same (or in a nearby) storage locations. Hence, the effect of splitting the customer orders is mitigated. For the storage policy, we observe that splitting the customer orders results in larger benefits when the SKUs are scattered in the warehouse (random assignment policy), which is the case of B2C e-commerce warehouses [66]. This result seems to be natural. Indeed, when we prohibit the splitting of the customer orders, we are forced to retrieve all the orderlines of a subset of customer order (batch of orders) in a single tour. Due to the random storage assignment policy, those orderlines are potentially stored in remote locations leading to long picking tours. Given a set of order batches, we can considerably reduce the picking time of those orders when allowing the splitting of customer orders by decomposing the order batches and assembling the nearby located orderlines of each batch in new picking tours. For the cart capacity, we observe a small growth in average savings with the increase of the cart capacity. An opposite effect is observed for the order structure where increasing the number of orders leads to a small reduction in order picking time. For the deadline distribution, no significant correlation between each factor level and its associated indicators (min., max., mean) can be observed. Globally, we can conclude that splitting the customer orders offers the best results for the configuration that corresponds to realistic e-commerce warehouses (large warehouse, random storage assignment policy, cart capacity of 45 orderlines). By restricting the analysis to instances with this configuration, the average reduction in order picking time reaches 49%.

From table 4.4, we also notice that $RFSS^{+}$ does not reduce the number of picking tours with respect to ILS. Given this observation, we conclude that the savings reported do not result from a better use of the cart capacity when splitting the customer orders. Thus, the

reduction in order picking time is essentially caused by the ability to split up a customer order over all the pickers and assemble closely located orderlines in a single tour leading to short picking tours.

Computation time analysis

We observe that although the complexity of our problem (the splitting of customer orders is allowed) is higher than the complexity of the problem introduced in [60] (the splitting of customer orders is avoided), $RFSS^{+}$ is considerably faster than ILS (56 s on average versus 123 s on average). Note that this observation does not prove that our heuristic is more efficient than ILS since the problems considered are different. However, we can conclude that the significant savings in order picking time reported in Table 4.4 do not come from higher computation efforts. Note also that in the implementation of our heuristic, we call the procedure `orderlinesBatching-SplitBasedProcedure`($\mathcal{O}^c$) for each cluster $\mathcal{O}^c$ sequentially. Since the clusters $\mathcal{O}^c \in \mathcal{C}$ are independent, significant reductions of the computational time could be obtained by calling the `orderlinesBatching-SplitBasedProcedure`($\mathcal{O}^c$) for each cluster in parallel.

The results show that the computational effort of our heuristic strongly depends on the size of the warehouse. the average CPU time increases from 34.8 s for small warehouses to 77.6 s for large warehouses. A same (but less significant) correlation is observed for the ILS algorithm (from 118.4 s to 127.1 s). For both algorithms, increasing the number of aisles and storage locations reduces the so-called "similarity of customer orders" (*i.e.*, the probability of finding orderlines of multiple orders in the same pick location) which makes the total number of pick locations in the warehouse higher and hence the TSPs more complex. For the storage assignment policy, we observe that the average computation time on instances that follow the "across aisles" storage policy (85.5 s) is considerably higher than the average computation time on instances that follow the "random" (43.6 s) and the "within aisle" (39.1 s) storage policies. The difference is also less significant for the ILS algorithm (139.5 s for "across aisles" storage policy and (114.3 s, 117.0 s) for the two other storage policies). We also observe a direct and positive correlation between the cart capacity and the average computation time. The increase in execution time is caused by the computation of the routing phase. Indeed, increasing the cart capacity leads to large clusters and hence, the computation time to execute the split procedure becomes higher. Note that the same correlation between the cart capacity and the execution time of the ILS algorithm is observed.

Finally, we observe that increasing the number of orders does not cause a significant increase in the average execution time unlike the ILS algorithm where the increase of orders number leads to much higher computational efforts. This result can be explained by the fact that the complexity of our algorithm is not related to the number of customer orders rather than the total number of orderlines to pick. The number of orderlines per order decreases with the growth of the number of orders (see section 4.5.1) which makes the growth in the total number of orderlines stable. This is not the case for the ILS algorithm where increasing the number of customer orders leads to an increase of the search space of the algorithm as explained in [60]. For the deadline distribution, the average execution times are uniformly distributed between the factor levels.

4.6 Conclusion

In this paper, we study the potential benefit of splitting the customer orders into orderlines during the picking process for warehouses that process a large number of small and time-critical orders (e-commerce warehouses). For this purpose, we extend the integrated batching, routing, and picker scheduling problem by allowing the orders to be splitted over several tours. The problem consists in determining a set of picking tours by regrouping orderlines, assigning the tours to a set of order pickers, and scheduling the tours assigned to each picker such that the orders deadlines are satisfied and the total processing time is minimized.

To solve the problem, we propose a route first-schedule second heuristic. In the routing phase, we divide the set of orders into clusters and apply a modified version of the split algorithm to determine the tours to retrieve the orderlines of each cluster. In the scheduling phase, we build a feasible scheduling of the picking tours over the order pickers by solving a constraint programming problem.

To assess the benefits of splitting the customer orders, we tested our method on a data set of 2970 instances with different orders characteristics and different warehouse environments generated by [60]. We compared the total order picking time of the solutions delivered by our heuristic with that of the solutions retrieved by the ILS of [60]. We found that by allowing order splitting, our heuristic delivers a massive reduction of 30% on average and 60% in the best case. For a planning horizon of 4 h, it represents more than a 1 h reduction of the shift of each picker on average with a maximum reduction of 1.8 h.

Our model assumes that the picking operations and the packing operations (*i.e.*, assemble the picked orderlines into final customer orders ready to be shipped) are planned sequentially. Since these operations appear to be correlated, further research should focus on extending our model to an integrated version that considers the picking and packing operations simultaneously. It would be worthwhile to model the several sorting systems existing in the e-commerce warehouses and compare them. Furthermore, we assume in our problem definition that the aisles are wide enought to safely neglect the effect of blocking on the performance of the picking process. This assumption may be too strong for small warehouses that are implanted in cities as a consequence of the growth of e-commerce caused by the COVID pandemic. One interessting reseach perspective would be to extend our model by considering the picker blocking as a hard constraint or a performance criterion.

CHAPITRE 5 ARTICLE 2 : THE PICKER ROUTING PROBLEM IN MIXED-SHELVES, MULTI-BLOCK WAREHOUSES

M. Haouassi, Y. Kergosien, J. Mendoza et L-M. Rouseau ont écrit cet article et l'ont soumis le 4 juillet 2022 au journal *Computers & Operations Research*

5.1 Introduction

In the last decades, the rapid development of information and communications technologies around the world has led to a drastic change in purchasing habits and the emergence of "e-commerce". Just to cite a few examples, Canadian e-commerce retail sales increased by 15.5% between 2017 and 2019; that is, an increment from 21.9 B$ in 2017 to 25.3 B$ in 2019 [4]. This trend was drastically accelerated by the COVID19 pandemic. During the early days of the pandemic, the governments of many countries applied various restrictions (*e.g.*, distancing measures, curfews, the closure of non-essential businesses, remote working) in an attempt to limit the rapid propagation of the virus. As a result, millions of locked down citizens massively turned towards online platforms to purchase goods that up to that point were retailed mostly through brick-and-mortar stores. For instance, Walmart e-commerce grocery business increased 74% during the pandemic [82]. The drastic increment in e-commerce transactions inevitably translated into an explosion of the number, complexity, and urgency of order fulfilment operations across industries.

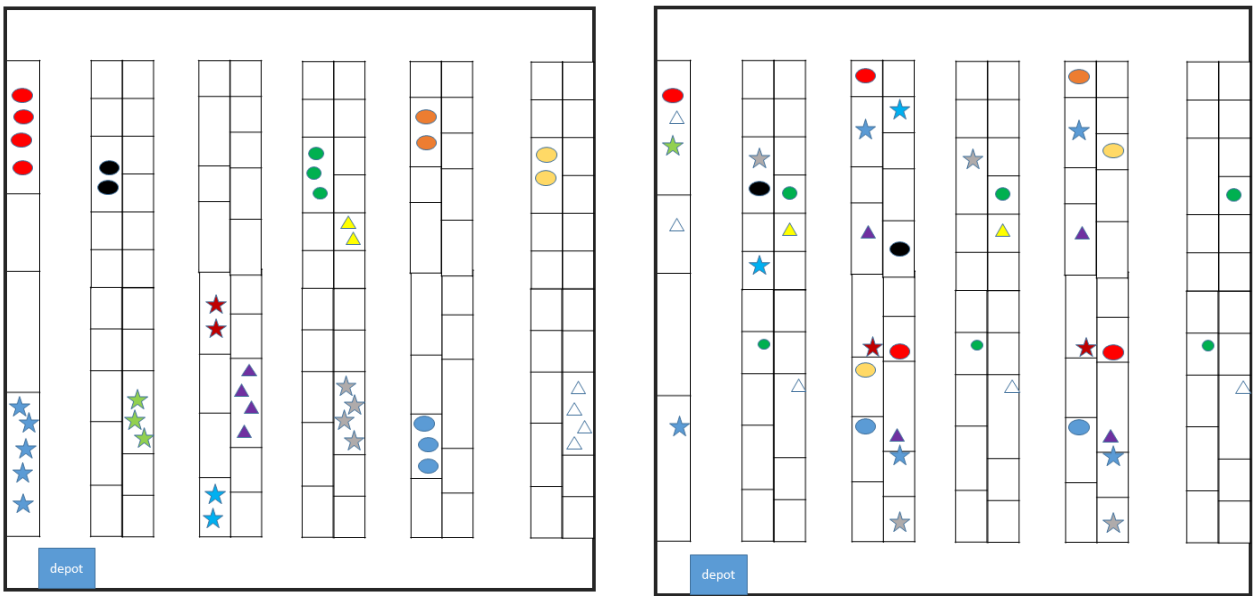
Contrary to classical warehouses, e-commerce *fulfillment centers* typically serve large numbers of small (in terms of both number of lines and number of units per line) orders that are time critical. Furthermore, to accommodate the wide range of orderlines, retailers must store a large assortment of items [66]. E-commerce fulfillment centers are usually smaller than traditional warehouses and are typically located in urban areas close to the main markets. Another notable difference between the two types of facilities are the policies implemented to assign items to storage positions. In traditional warehouses, each stock keeping unit (SKU) is assigned to a dedicated shelf (see Figure 5.1a). Under this policy, the most "popular" SKUs (*i.e.*, SKUs with high demand frequency) can be stored close to the sorting and packing stations (hereafter referred to as the depots) [13]. On the other hand, e-commerce fulfillment centers tend to implement what is commonly known as a mixed-shelves policy. As the name suggests, under this policy, each storage position may hold one or more units of multiple SKUs. Needless to say, the inventory for one SKU can be spread throughout multiple loca-

tions around the warehouse (see Figure 5.1b). The mixed-shelves policy allows for a better use of the storage space, because fresh inventory for a given SKU may be stored at any empty shelf. Furthermore, studies suggest that this policy minimizes the average travel time in the warehouse [5].

E-commerce warehouses implement several picking systems. They usually follow two paradigms: *picker-to-part* and *part-to-picker*. In a *picker-to-part* systems, (human) order pickers move in the picking area to retrieve the items from static shelves (storage locations) and bring them to the packing station. In a *part-to-picker* systems, a fleet of robots bring mobile storage units (shelves/baskets) to the picking station where human operators pick the required items. In this paper, we focus on the first system which is the most widely used in practice [1]. Needless to say, moving between storage locations is time consuming and labor intensive [47]. Hence, minimizing the number and distance of these movements is key to achieve supply chain efficiency.

The majority of the picking literature is dedicated to optimizing the picking process in classical warehouses. The most studied problem is known as the picker routing problem (PRP). The problem is defined as follows: given a central depot and a *pick list* that consists of a set of SKUs to pick from their corresponding storage locations (shelves), find the shortest tour that starts from the depot, visits all the picking locations and ends at the depot. This problem can be modeled as a symmetric traveling salesman problem (TSP) where the node set is made up of the depot and the storage locations and the edge set is the set of shortest paths connecting the nodes though the warehouse aisles. TSP methods are therefore natural candidates to solve PRPs in practice. However, the community has been very active in developing problem-specific methods that exploit the rectilinear distance matrix induced by the particular layout of warehouses.

Most exact methods rely on the seminal work of Ratliff and Rosenthal in [27]. They established that the set of possible ways (transitions) to cover an aisle in an optimal solution is limited to 6, regardless of the number of storage locations to visit in this aisle. Based on this result, they proposed a dynamic programming (DP) algorithm that solves the PRP for 1-block warehouses (*i.e.*, a warehouse with 2 cross aisles). In [11], Roodbergen and De Koster extended the results in [27], to 2-block warehouses. More recently, Pansart et al. [29] proposed a DP algorithm to solve the PRP for k-block warehouses. Their work is based on that of Cambazard et al. [31] which solved the rectilinear-TSP. Finally, Schiffer et al. [10] studied a more general picking problem in which the classical PRP appears as a subproblem. They



(a) Traditional assignment policy (b) Mixed-shelves policy

Figure 5.1 1-block warehouse with 5 aisles and 2 cross aisles

proposed a bidirectional layered graph algorithm for solving the PRP for k-block warehouses and showed that their approach outperforms the DP of [29] by 67%. Furthermore, they propose a competitive integer linear programming formulation (ILP) to solve the PRP.

The literature also reports on simple and intuitive policies to route pickers inside the warehouses. Probably the two best known are the S-shape and the Return policies. In the former, the picker entirely traverses every aisle (but the last one) storing an item that is included on his or her picking list. In the latter, the picker enters and exits every such aisle from the same point [83]. While these policies are easy to implement in the warehouse floor, they may lead to far-from-optimal solutions. For instance, according to Theys et al. [32], exact methods can deliver improvements of up to 47% (in terms of total travel distance) with respect to these simple policies. Furthermore, in today's warehouses, more often than not, the picking operations are supported by technology such as pick-to-voice, that facilities the implementation of the less intuitive picking tours delivered by exact methods. Therefore, there is little incentive not to try to exploit exact methods in practice.

The picker routing problem in mixed-shelves warehouses (PRP-MS) differs from the classical PRP. In the PRP-MS, the location(s) from where each SKU is to be retrieved is selected before designing the tour. Few articles study this problem. [34] proposes fast heuristics that

use selection rules to choose the locations from where to pick the SKUs and then design the tour that visits the selected locations. To the best of our knowledge, the work of Goeke et al. [35] is the only paper that tackles the PRP-MS with an exact method. They propose an ILP formulation that solves the classic PRP by exploiting the characteristics of an optimal solution introduced in [27]. Then, they adapt their ILP to accommodate other features of e-commerce warehouses, such as: mixed-shelves storage, decoupling of picker and cart, and multiple end depots. However, their work focuses on 1-block warehouses and cannot be extended to the multi-block layout.

In this paper, we investigate the PRP-MS in multi-block warehouses. We propose a logic-based Benders decomposition (LBBD) procedure to solve the problem. The main contributions of the paper are:

— Proposing the first exact method to solve the PRP-MS in multi-block warehouses.
— Introducing several improvements to the proposed algorithm including a lower bounding function and valid inequalities, and providing a detailed analysis of their effectiveness.
— Adapting the state-of-art ILP for the classical PRP to work on mixed-shelves warehouses. The resulted ILP is named `SCH-MS+`.
— Showing through extensive computational experiments on both benchmark data from the literature (1-block) and new instances (multi-block) the nifty performance of the proposed method compared to `SCH-MS+`.

The remainder of this paper is organized as follows: in Section 2, we formally describe the extension of PRP-MS to the multi-block layout. In Section 3, we present the LBBD procedure. In Section 4, we detail the proposed algorithmic improvements. In Section 5, we present the SCH-MS. In Section 6, we report computational results. Finally, we provide conclusions and outline research perspectives in Section 7.

5.2 Problem Statement

The picker routing problem in mixed-shelves warehouses is defined in a warehouse with the following characteristics:

— **a multi-block layout,** composed of **"picking aisles"** and **"cross aisles"**. The picking aisles allow the picker to access the shelves (storage locations) and retrieve SKUs. The cross aisles divide the aisles into **"subaisles"** and the picking area into blocks. From a cross aisle the picker can enter or exit a subaisle, or move to other

aisles. All the subaisles have the same length, as well as the cross aisles.

— **a manual picker-to-part system,** in which an order picker starts from a central **depot** (*e.g.*, a sorting and packing station), walks through picking and cross aisles, stops at **picking locations** to retrieve SKUs from shelves, and comes back to the depot. Each picking location is mapped to one or more accessible storage locations. During a picking tour, the picker processes a single order (pick by order) or a compilation of orders (pick by batch). In both cases, he or she is guided by a "picklist" in which the requested SKUs and their associated quantities are listed. Figure 5.2 depicts an example of a 2-block warehouse with the associated picking locations and the storage locations accessible from each picking location.

— **a mixed-shelves storage policy,** where unit loads of an SKU are broken down into single items, that are randomly spread throughout the warehouse. Hence, each SKU is available in multiple storage locations and each storage location can handle multiple SKUs.

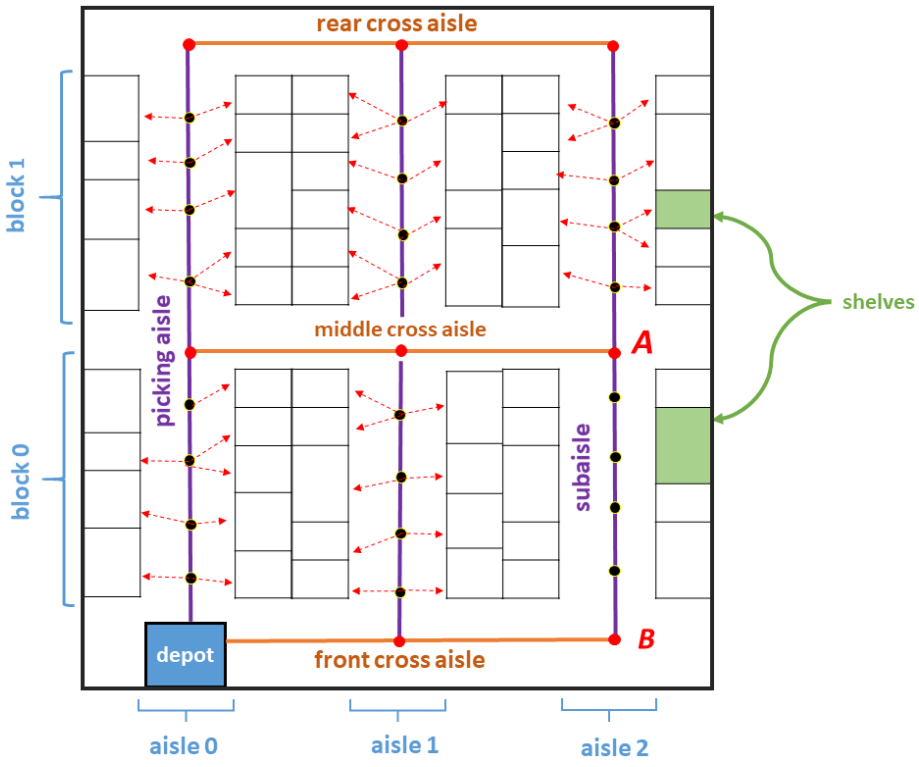


Figure 5.2 Representation of 2-block warehouse

Formally, let us define $\mathcal{L} = \{0, \cdots, L-1\}$ as the set of picking aisles and $\mathcal{K} = \{0, \cdots, K-1\}$ as the set of blocks in the warehouse. The pair (k, l) denotes the subaisle located at block k and picking aisle l. For simplicity, we assume that the depot is located at the bottom left corner node of the graph (see Figure 5.2). We define $t_{(k,l)}$ and $b_{(k,l)}$ as the top and bottom nodes of subaisle (k, l). For instance, in the example depicted in Figure 5.2, $t_{(0,2)} = A$ and $b_{(0,2)} = B$.

We consider a picklist $\mathcal{I} = \{1, \cdots, s, \cdots, S\}$ in which the SKUs to pick in a single picking tour are itemized. Each SKU $s \in \mathcal{I}$ is requested in r_s units, and is available at a set of picking locations $\mathcal{P}_s$. For each $p \in \mathcal{P}_s$, we define q_p^s as the quantity of s available at the location. Let $\mathcal{P}$ be the set representing the depot and all the picking locations. Finally, we define $d_{p,p'}$ as the shortest distance between locations p and $p' \in \mathcal{P}$.

The objective of the problem is to select the picking locations from where to pick the SKUs and design a minimal-length tour that starts from the depot, visits the selected picking locations, and comes back to the depot. Note that due to the limited quantities of SKUs stored in each storage location, the picker could visit multiple picking locations to satisfy the demand of each SKU. Let $\pi = (0, 1, \cdots, p, \cdots, P, 0)$ be a solution representation of our problem in which 0 represents the depot and the other indices represent the selected picking locations sorted according to the visiting order. Solution π is feasible if the sum of the quantities of each SKU retrieved from the selected storage locations cover all the requests in the picking list $\mathcal{I}$. Formally, the following equation must be satisfied:

$$\sum_{p \in \pi | s \in \mathcal{I}} q_p^s \geq r_s \cdot \tag{5.1}$$

Let $f(\pi)$ be the function that returns the total travel distance of π. We seek for π^{opt}, the solution that minimizes the total distance traveled:

$$\pi^{opt} = argmin_\pi \left(f(\pi) = d_{0,1} + \sum_{p=1}^{P-1} d_{p,p+1} + d_{P,0} \right) \cdot \tag{5.2}$$

Note that in our problem description, the cart capacity is assumed to be sufficient (it can carry all the items in the picklist) and that the search and pick time at each picking location is negligible. These assumptions are consistent with those commonly adopted in the PRP-MS literature [34, 35].

5.3 Logic-based Benders Decomposition

Logic-based Benders decomposition is a generalization of the classical Benders decomposition procedure where the subproblem cannot be dualized using linear programming duality. Instead, LBBD derives Benders cuts by solving a more general inference dual. The solution of the inference dual provides the tightest bound on the cost of the master problem implied by its current solution. This bound is then used to generate cuts that are added to the master problem. Note that no systematic procedure is available to derive such cuts, as in the case of classical Benders decomposition. Therefore, cuts must be tailored to the problem at hand [84].

LBBD has been successfully applied to scheduling [85, 86], routing [87], and location [88] problems. In our case, the main motivation behind using logic-based Benders decomposition is that once the picking locations are selected and fixed, we can leverage a number of powerful algorithms from the literature to solve the remaining part of the problem, that is, the classical PRP. We thus decompose the full problem into a master problem and a subproblem. The former deals with the picking location selection decisions, while the latter deals with the routing decisions. In the following, we describe the master problem, the subproblem, and the Benders cuts that we generate. For the sake of readability, we shift the proof of the validity of the Benders cuts to Appendix B.1.

5.3.1 Master problem

Let $\mathcal{C}_{k,l}$ be the set of non-dominated configurations of subaisle (k, l). Each configuration $c \in \mathcal{C}_{k,l}$ represents a subset $\mathcal{P}^c_{k,l}$ of picking locations that results from traversing the subaisle (k, l), entering and exiting the subaisle (k, l) from the top (sloping from the top), entering and exiting the subaisle (k, l) from the bottom (sloping from the bottom), or entering and exiting the subaisle (k, l) from the top and bottom (sloping from both sides). The construction of $\mathcal{C}_{k,l}$ is based on the following observations:

- If we completely traverse a subaisle (k, l) that contains SKUs to pick, it is always feasible and profitable to pick all the SKUs that can be picked in this subaisle. This is due to the fact that the search and pick time at each picking location are assumed to be 0.
- More generally, if the picker enters a subaisle (k, l) from $t_{(k,l)}$ (resp. $b_{(k,l)}$), walks up to a location p (resp. p'), and comes back to $t_{(k,l)}$ (resp. $b_{(k,l)}$), it is always feasible and profitable to pick all SKUs that can be picked along the way between $t_{(k,l)}$ and p (resp. between $b_{(k,l)}$ and p').

We thus eliminate multiple dominated configurations. Figure 5.3 gives an example of a subaisle that contains 4 shelves in both sides and the 10 possible configurations associated with this subaisle. For each configuration, the black nodes represent the locations at which the picker stops to pick items. The orange nodes represent the top and the bottom of the subaisle. The last two configurations represent an example of 2 dominated configurations that are eliminated.

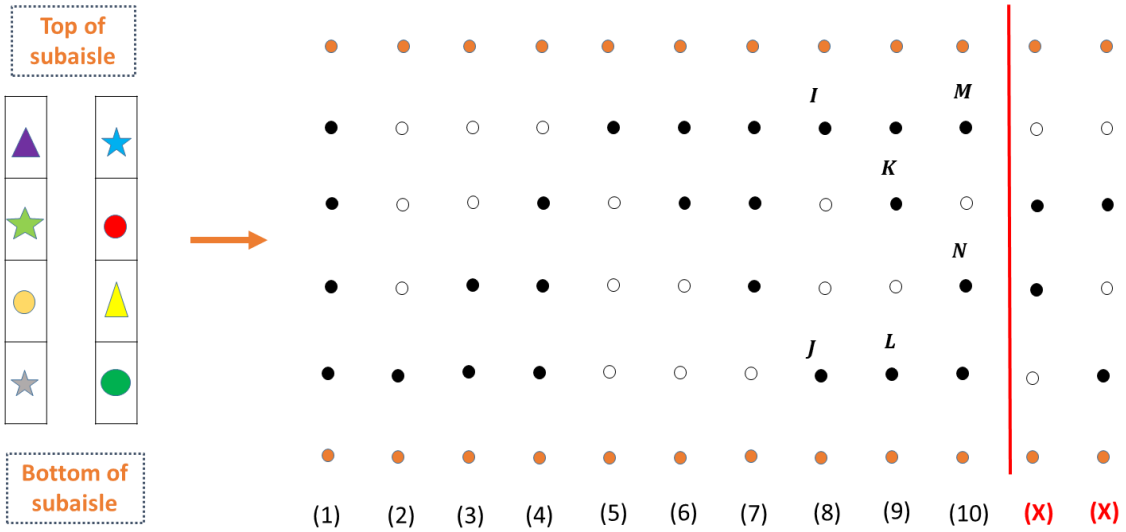


Figure 5.3 Set of non-dominated configurations and an example of dominated ones for a subaisle with 4 locations

In addition to the previously defined parameters, our master problem formulation uses the following parameter and variables:

— q_c^s: a parameter denoting the number of SKUs s that can be retrieved using configuration c.

— $x_{k,l}^c$: a binary variable that takes the value of 1 if configuration c is selected for subaisle (k, l), 0 otherwise.

— θ: a non-negative variable that captures the lower bound on the total distance traveled in the tour.

Finally, the master problem is given by:

$$\min \theta \tag{5.3}$$

$$\sum_{c \in \mathcal{C}_{k,l}} x^c_{k,l} \leq 1 \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \tag{5.4}$$

$$\sum_{k=0}^{K-1} \sum_{l=0}^{L-1} \sum_{c \in \mathcal{C}_{k,l}} q^s_c \cdot x^c_{k,l} \geq r_s \qquad \forall s \in \mathcal{I} \tag{5.5}$$

$$\theta \geq \qquad \textit{bounding functions of optimality cuts} \tag{5.6}$$

$$x^c_{k,l} \in \{0,1\}, \quad \theta \in \mathbb{R}^+ \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L}, \forall c \in \mathcal{C}_{k,l} \tag{5.7}$$

The objective function (5.3) minimizes the lower bound θ. Constraints (5.4) ensure that at most one configuration is selected for each subaisle (k, l). The request of each SKU is satisfied with constraints (5.5). Constraints (5.6) are optimality cuts generated in the previous iterations and will be discussed in the next subsection. Finally, constraints (5.7) define the nature of decision variables.

5.3.2 Subproblem and optimality cuts

Let $\bar{\mathcal{C}}$ be the set of configurations selected in the master problem ($\bar{\mathcal{C}} = \{(c, k, l) : c \in \mathcal{C}_{k,l}$ & $x^c_{k,l} = 1\}$). Furthermore, let $\bar{\mathcal{P}}$ be the set of picking locations associated with the selected configurations ($\bar{\mathcal{P}} = \{p \in \mathcal{P}^c_{k,l} : (c, k, l) \in \bar{\mathcal{C}}\}$). The subproblem consists in finding the shortest tour that visits all the picking locations in $\bar{\mathcal{P}}$. Note that for any $\bar{\mathcal{P}} \subseteq \mathcal{P}$, the subproblem is feasible. Hence, only optimality cuts are generated from the subproblem. Our Benders cuts must satisfy the following properties (adapted from [89]):

1. The cut must exclude the current master problem solution if it is not globally optimal.
2. The cut must not remove any globally optimal solution.

Given these properties and assuming that $\bar{\theta}$ is the total distance of the optimal tour that visits the picking locations in $\bar{\mathcal{P}}$, we define the following optimality cut:

$$\theta \geq \bar{\theta} - \sum_{(c,k,l) \in \bar{\mathcal{C}}} \sum_{p \in \mathcal{P}^c_{k,l}} \left[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p} \right] \tag{5.8}$$

5.3.3 LBBD implementation

Algorithm 5.1 describes the general structure of our LBBD procedure. First, the upper bound UB, the lower bound LB, and an iteration counter i are respectively set to ∞, 0, and 0 in the initialization phase. The body of the procedure (lines $2 - 9$) consists in repetitively

solving the master problem, the subproblem, and updating UB and LB until the stopping criterion is reached (*i.e.*, $|(UB - LB)/UB| <= \epsilon$). At each iteration i, the algorithm solves the master problem with the previously generated optimality cuts (Line 3), generates $\bar{\mathcal{C}}^i$ (the set of selected configurations), and sets LB to the objective value of the master problem. Then, it solves the picker routing problem with the picking locations included in $\bar{\mathcal{C}}^i$ and returns $\bar{\theta}^i$, the distance of the optimal tour. Next, it constructs an optimality cut which is added to the master problem. Finally, it updates UB depending on $\bar{\theta}^i$ and proceeds to the next iteration.

Algorithm 5.1 LBBD procedure: general structure

1: $UB \leftarrow \infty, LB \leftarrow 0, i \leftarrow 0$

2: **while** $|(UB - LB)/UB| < \epsilon$ **do**

3: Solve the master problem and obtain $\bar{\mathcal{C}}^i$ (the selected configurations) and θ^i (the value of θ at iteration i)

4: $LB \leftarrow \theta^i$

5: Solve the subproblem and obtain $\bar{\theta}^i$ (the optimal distance)

6: Add the following cut to the master problem

$$\theta \geq \bar{\theta}^i - \sum_{(c,k,l)\in\bar{\mathcal{C}}^i} \sum_{p\in\mathcal{P}^c_{k,l}} \left[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p}\right]$$

7: $UB \leftarrow min(UB, \bar{\theta}^i)$

8: $i \leftarrow i + 1$

5.4 Enhancing the LBBD decomposition method

A naive implementation of Benders decomposition may suffer from several drawbacks (*e.g.*, ineffective initial iterations, loose cuts) leading to excessive computational times. Researchers have developed a number of techniques to accelerate Benders decomposition-based methods (such as LBBD). These techniques are usually intimately related to the problem being studied. They include: valid inequalities, upper or lower bounding, and warm starting. For a detailed discussion of these techniques, we refer the reader to [84].

In our preliminary experiments, we observed that our base-line LBBD suffers from ineffective initial iterations. Indeed, the master problem returns successive solutions without increasing the lower bound. This behavior is due to the loss of the information provided by subproblem's variables and constraints. To address this drawback, we developed two accel-

eration techniques tailored to our problem: a lower bounding function (LBF), and a set of valid inequalities (VIs).

5.4.1 Lower bounding function

The intuition behind our LBF is to provide the master problem with a lower bound on the total distance of the future picking tour given a set of selected configurations. Our LBF is equal to the sum of the *horizontal* distance and the *inside-subaisles* distance.

The horizontal distance corresponds to a roundtrip between the depot and the farthest visited aisle. To compute it, we first define for each aisle $l \in \mathcal{L}$, $\bar{d}_l$ as the shortest distance between aisle l and the depot. Then, we introduce a non-negative variable λ that captures the horizontal distance and add the following constraint to the master problem:

$$\lambda \geq 2 \cdot \sum_{c \in \mathcal{C}_{k,l}} \bar{d}_l \cdot x^c_{k,l} \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \tag{5.9}$$

The inside-subaisles distance is defined as the sum of the approximated distances associated with each subaisle. To compute it, we first define $\tilde{d}^c_{k,l}$ as the approximated distance associated with configuration $c \in \mathcal{C}_{k,l}$. We propose two methods to compute $\tilde{d}^c_{k,l}$. The first one is based on the following proposition adapted from [27]:

Proposition 1. *Let $\bar{\mathcal{C}}$ be the set of configurations selected in the master problem. An optimal tour that visits all locations in $\bar{\mathcal{P}}$ contains only six possible transitions of a subaisle (k, l).*

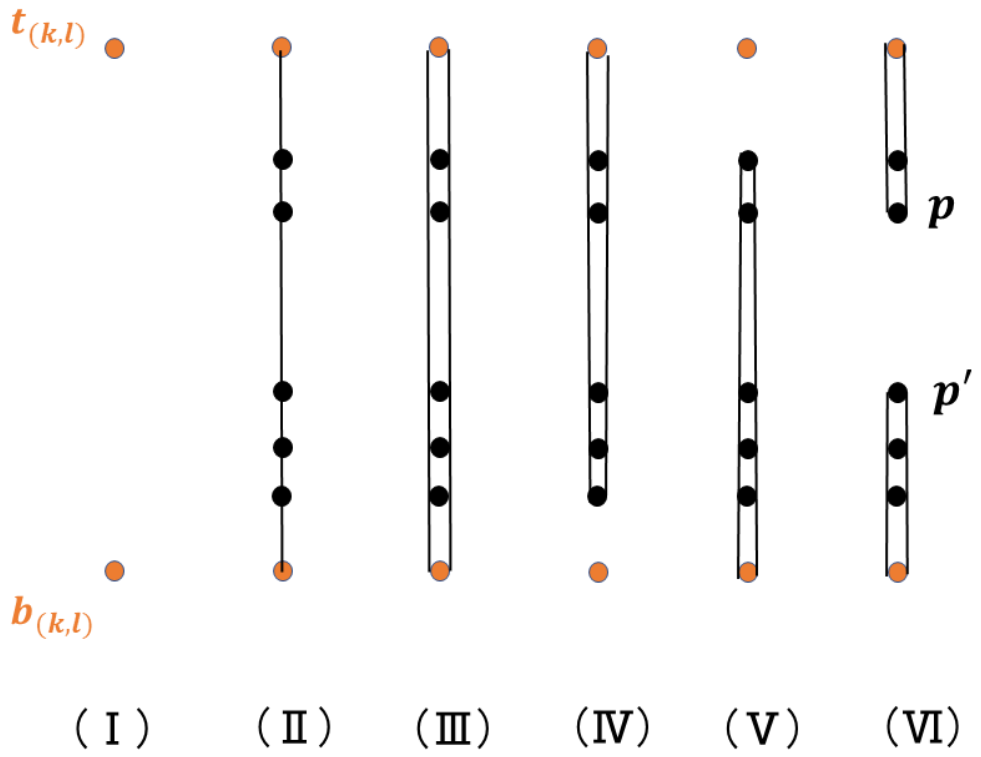


Figure 5.4 Possible subaisle transitions for an optimal path

Let us assume that configuration c is selected for the subaisle (k, l) in the master problem. Figure 5.4 shows an example of the possible transitions of (k, l) with five picking locations. It includes:

(I) no-traversal: the picker does not traverse subaisle (k, l), in other words, no configuration has been selected for subaisle (k, l)

(II) complete traversal: the picker completely traverses the subaisle (k, l)

(III) two-direction traversal: the picker completely traverses the subaisle (k, l) in both directions

(IV) top slope: the picker enters subaisle (k, l) from $t_{(k,l)}$, walks to the farthest location $\in \mathcal{P}^c_{k,l}$, and comes back to $t_{(k,l)}$

(V) bottom slope: the picker enters subaisle (k, l) from $b_{(k,l)}$, walks to the farthest location $\in \mathcal{P}^c_{k,l}$, and comes back to $b_{(k,l)}$

(VI) double slope: the picker enters subaisle (k, l) from $t_{(k,l)}$ and $b_{(k,l)}$, walks to locations p and p', and comes back to $t_{(k,l)}$ and $b_{(k,l)}$, respectively. p and p' correspond to the two furthest successive locations $\in \mathcal{P}^c_{k,l}$

We define $\tilde{d}^c_{k,l}$ as the shortest distance that can be traveled to cover the subaisle (k,l) with configuration c. It corresponds to the minimum distance associated with transitions (II,IV,V,VI).

To model the LBF, we introduce for each subaisle (k,l), a non-negative variable $\psi_{k,l}$ that equals the approximated distance to travel in this subaisle, and add the following constraints to the master problem:

$$\psi_{k,l} \geq \sum_{c \in \mathcal{C}_{k,l}} \tilde{d}^c_{k,l} \cdot x^c_{k,l} \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \tag{5.10}$$

$$\theta \geq \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} \psi_{k,l} + \lambda \tag{5.11}$$

5.4.2 Improving the LBF

We observed in our preliminary experiments that multiple configurations $c \in \mathcal{C}_{k,l}$ have the same approximated distance. Thus, the master problem may return equivalent solutions in successive iterations leading to a short term stagnation of the lower bound. To reduce this negative effect, we propose a second method to compute the approximated distances $\tilde{d}^c_{k,l}$.

Let us divide $\mathcal{C}_{k,l}$ into four subsets: $\check{\mathcal{C}}_{k,l}$ the set of configurations in which the subaisle is entered from $t_{(k,l)}$; $\hat{\mathcal{C}}_{k,l}$ the set of configurations in which the subaisle is entered from $b_{(k,l)}$; $\check{\hat{\mathcal{C}}}_{k,l}$ the set of configurations in which the subaisle is entered from both $b_{(k,l)}$ and $t_{(k,l)}$; and $\mathcal{C}^{all}_{k,l}$ a singleton set including the configuration that contains all picking locations of subaisle (k,l). In the example depicted in Figure 5.3, $\check{\mathcal{C}}_{k,l}$ includes configurations $(5,6,7)$, $\hat{\mathcal{C}}_{k,l}$ includes configurations $(2,3,4)$, $\check{\hat{\mathcal{C}}}_{k,l}$ includes configurations $(8,9,10)$, and $\mathcal{C}^{all}_{k,l}$ includes only configuration (1). The approximated distance for each configuration $c \in \mathcal{C}_{k,l}$ is computed as follows:

- if $c \in \check{\mathcal{C}}_{k,l}$, $\tilde{d}^c_{k,l}$ equals to the distance covered by entering from $t_{(k,l)}$, walking to the farthest picking location $\in \mathcal{P}^c_{k,l}$, and coming back to $t_{(k,l)}$
- if $c \in \hat{\mathcal{C}}_{k,l}$, $\tilde{d}^c_{k,l}$ equals to the distance covered by entering from $b_{(k,l)}$, walking to the farthest picking location $\in \mathcal{P}^c_{k,l}$, and coming back to $b_{(k,l)}$
- if $c \in \check{\hat{\mathcal{C}}}_{k,l}$, $\tilde{d}^c_{k,l}$ equals $2 \times$ (the distance between $t_{(k,l)}$ and p^t plus the distance between $b_{(k,l)}$ and p^b), where p^t and p^b are defined such that: all the locations in $\mathcal{P}^c_{k,l}$ are positioned between $t_{(k,l)}$ and p^t, and between $b_{(k,l)}$ and p^b; all the locations in $\mathcal{P}_{k,l} \setminus \mathcal{P}^c_{k,l}$ are positioned between p^t and p^b. For simplicity, we refer to p^t and p^b as the *borders* of configuration c. An example of these borders is given in Figure 5.3. p^t is represented

with the nodes at positions $8I$, $9K$, and $10M$ while p^b is represented with the nodes at positions $8J$, $9L$, and $10N$.

— if $c \in \mathcal{C}^{all}_{k,l}$, $\tilde{d}^c_{k,l}$ equals the shortest distance returned by the first computation method.

Proposition 2. *Computing the approximated distance of each configuration $c \in \mathcal{C}_{k,l}$ using the second method does not remove any optimal tour*

Proof. Let us assume that in the optimal solution, all the locations in $\mathcal{P}^c_{k,l}$ must be visited and the optimal tour covers subaisle (k,l) with a distance $d^{opt}_{(k,l)}$ smaller than $\tilde{d}^c_{k,l}$. Then, we distinguish the following cases:

1. $c \in \check{\mathcal{C}}_{k,l}$: $d^{opt}_{(k,l)}$ is given by one of the following three transitions: a complete traversal, a bottom slope, a double slope. Since the picking locations in $\mathcal{P}_{k,l} \setminus \mathcal{P}^c_{k,l}$ are positioned at the bottom of the picking locations in $\mathcal{P}^c_{k,l}$, the picker will necessarily visit all the picking locations in subaisle (k,l) when using one of the three alternative transitions. However, there is another configuration $c' \in \mathcal{C}^{all}_{k,l}$ that visits all the picking locations of subaisle (k,l) (*i.e.*, $\mathcal{P}^c_{k,l} \subset \mathcal{P}^{c'}_{k,l}$), which associated distance $\tilde{d}^{c'}_{k,l}$ is the shortest one (*i.e.*, the distance computed with the first method). Given that we aim at minimizing the value of θ and no cart capacity is assumed in the problem definition, the master problem will necessarily select configuration c'. An example of this situation is illustrated in Figure 5.5 for a subaisle with four picking locations. $\tilde{d}^c_{k,l}$ and $\tilde{d}^{c'}_{k,l}$ are highlighted by orange dashed lines. In addition, the shortest distance to visit the locations in $\mathcal{P}^c_{k,l}$ corresponds to a complete traversal (red line).

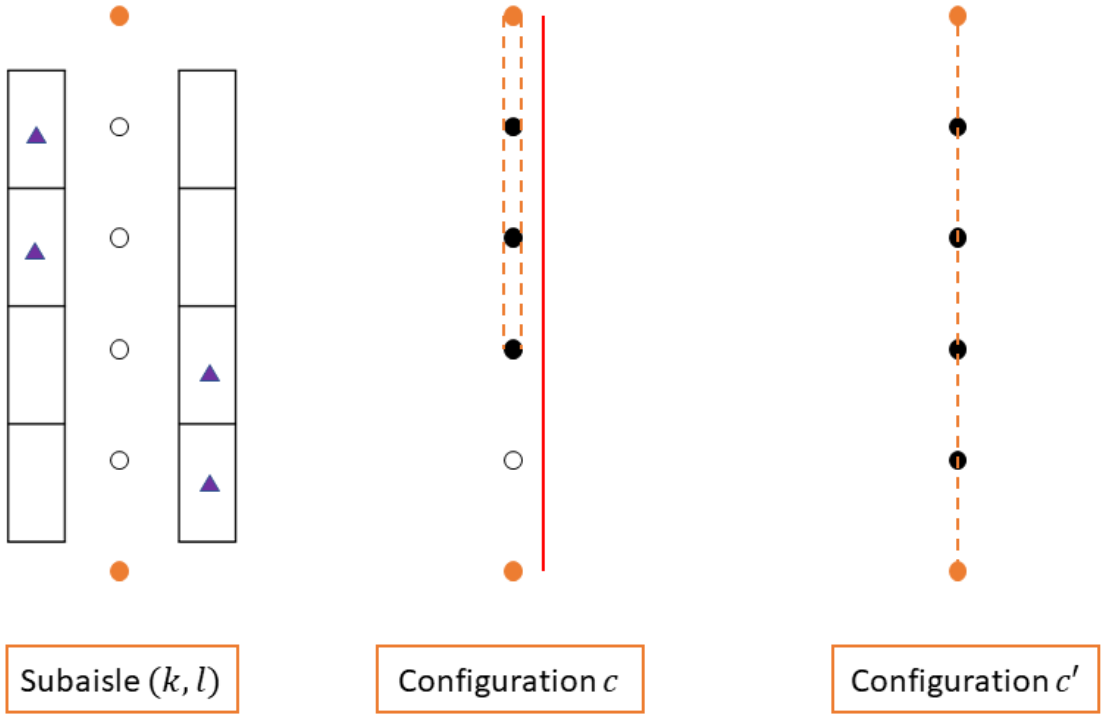


Figure 5.5 An illustrative example of the first case

2. $c \in \hat{\mathcal{C}}_{k,l}$: $d^{opt}_{(k,l)}$ is given by one of the three following transitions: a complete traversal, a top slope, a double slope. Note that the picker necessarily visits all the picking locations in subaisle (k, l) if he or she uses one of the three alternative transitions. Then, we can use the previous argument to prove that no optimal solution is excluded.
3. $c \in \check{\hat{\mathcal{C}}}_{k,l}$: $d^{opt}_{(k,l)}$ is given by one of the four following transitions: a complete traversal, a top slope, a bottom slope, or a double slope different from the one associated with $\tilde{d}^{c}_{k,l}$. In all these transitions, the picker necessarily visits all the picking locations of subaisle (k, l). Again, the previous argument can be applied to prove that no optimal solution is excluded. Figure 5.6 shows an example of this situation for a subaisle with four picking locations and $\mathcal{P}^{c}_{k,l}$ defined by three locations. $\tilde{d}^{c}_{k,l}$ and $\tilde{d}^{c'}_{k,l}$ are highlighted by orange dashed lines. In this case, the shortest distance to visit the locations in $\mathcal{P}^{c}_{k,l}$ corresponds to a different double slope (red lines).

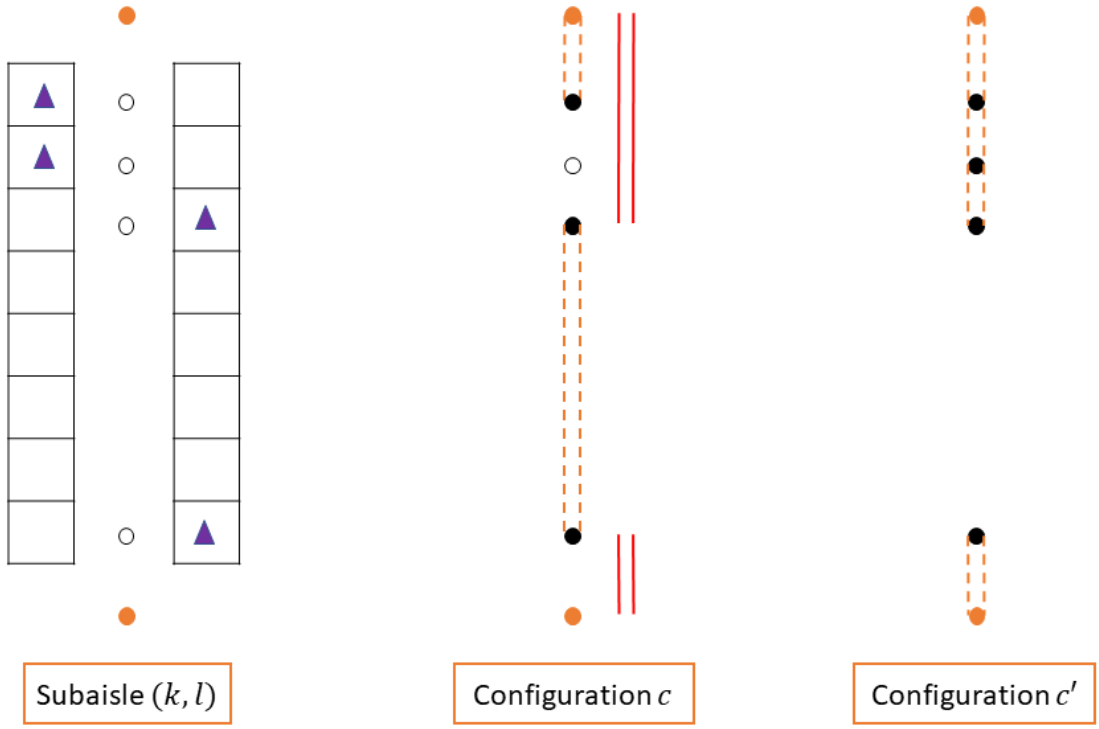


Figure 5.6 An illustrative example of the third case

□

From the second computation method, one might have the intuition to simply remove every configuration c from $\mathcal{C}_{k,l}$ if there exists another configuration c' such that $\mathcal{P}^c_{k,l} \subset \mathcal{P}^{c'}_{k,l}$ and $\tilde{d}^c_{k,l} > \tilde{d}^{c'}_{k,l}$. However, by doing so, we may remove an optimal solution to the problem. Figure 5.7 depicts an example of this situation for a 1-block warehouse with two aisles. We assume that only the purple SKUs must be retrieved. Let c be the configuration of the first aisle that includes the locations represented with black nodes. Its associated distance $\tilde{d}^c_{k,l}$ is represented with two red lines. We observe that there is another configuration c' that contains all the picking locations, with an associated distance $\tilde{d}^{c'}_{k,l}$, that is shorter than $\tilde{d}^c_{k,l}$. However, the tour resulting from the selection of c in the first aisle (*i.e.*, the tour represented with continuous lines) is smaller than the tour that results from the selection of c' (*i.e.*, the tour represented with dashed lines).

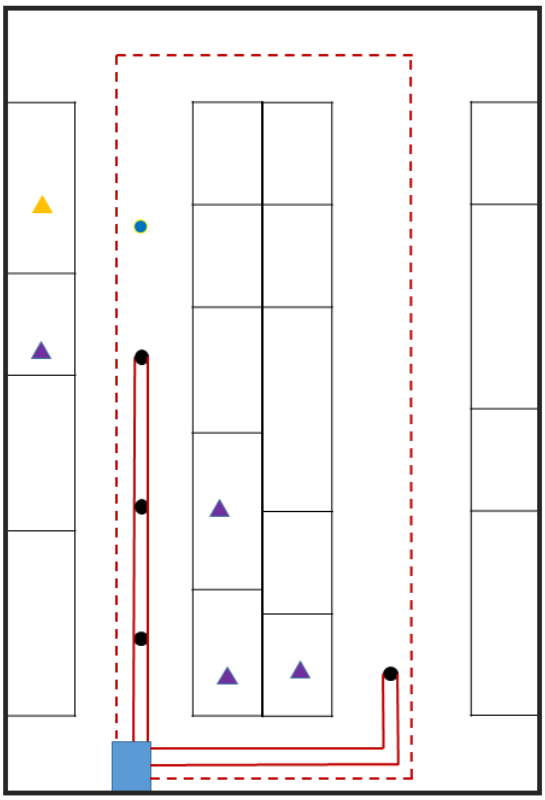


Figure 5.7 An illustrative example to show the importance of keeping all the configurations associated with a subaisle

5.4.3 Valid inequalities

In this part, we introduce two sets of valid inequalities for the master problem. The first set works on each block $k \in \mathcal{K}$ while the second set works on each pair of successive blocks $(k, k+1), k \in \{0, \cdots, K-2\}$. The objective of these VIs is to eliminate poor-quality solutions and increase the value of the LBF.

Intra-block VIs

The intra-block VIs are based on the following observation: for a given block k, if subaisle (k, l) is covered with a top slope configuration c (*i.e.*, $c \in \check{\mathcal{C}}_{k,l}$), then there are at least two other subaisles in block k that must be completely traversed. Since we cannot enforce complete traversal of two subaisles in the master problem without introducing big-M constraints, we approximate these decisions by selecting configurations $c_1 \in \mathcal{C}^{all}_{k,l_1}$ and $c_2 \in \mathcal{C}^{all}_{k,l_2}$ for two subaisles (k, l_1) and (k, l_2). The intra-block VIs are modeled as follows:

$$2 \cdot \sum_{c \in \bar{\mathcal{C}}_{k,l}} x^c_{k,l} \leq \sum_{l'=0, l' \neq l}^{L-1} \sum_{c \in \mathcal{C}^{all}_{k,l'}} x^c_{k,l'} \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \qquad (5.12)$$

Figure 5.8 illustrates an intra-block VI for a 1-block warehouse. The warehouse includes three subaisles $(0,0)$, $(0,1)$, and $(0,2)$. The subaisles $(0,0)$ and $(0,1)$ contain three picking locations while subaisle $(0,2)$ contains two. In subaisle $(0,1)$, the configuration c is highlighted with black nodes and its approximated distance $\tilde{d}^c_{0,1}$ is drawn in red lines. If this configuration is selected, then $c_1 \in \mathcal{C}^{all}_{0,0}$ and $c_2 \in \mathcal{C}^{all}_{0,2}$ must be selected for subaisles $(0,0)$ and $(0,2)$ which is shown in Figure 5.8 with black nodes for the configurations to select, and with orange dashed lines for the approximated distance associated with each selected configuration.

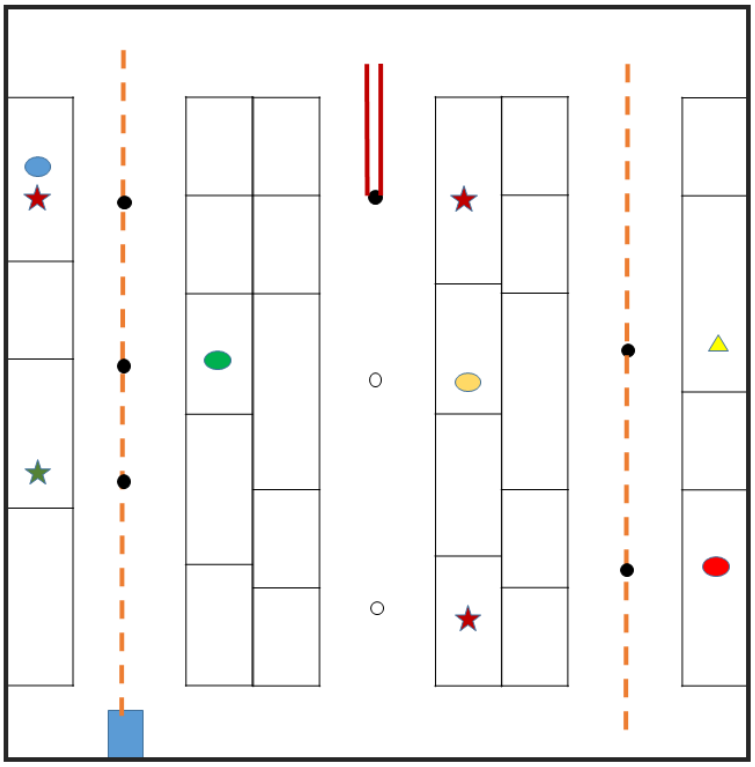


Figure 5.8 An illustrative example for intra-block VIs

Inter-block VIs

The idea behind our inter-block VIs is to link two consecutive blocks k and $k+1$ with a constraint that forces block k to be traversed twice if block $k+1$ is reached. Contrary

to intra-block VIs, we distinguish for the inter-block VIs two cases: In the first case, the block $k+1$ must only be reached in its first subaisle (belonging to the aisle containing the depot). Hence, we may have a configuration in which the block k is only traversed twice in its first subaisle. In the second case, the block $k+1$ is reached in an aisle $l > 0$. In this case, the block k may be traversed twice in two distinct subaisles without removing the optimal solution. Since we cannot force the subaisles in block k to be completely traversed in the master problem, we force the selection of one configuration $c \in \mathcal{C}^{all}_{k,l}$ (for subaisle (k,l)) or two configurations $c_1 \in \mathcal{C}^{all}_{k,l_1}$ and $c_2 \in \mathcal{C}^{all}_{k,l_2}$ (for subaisles (k,l_1) and (k,l_2)) depending on the first or second case, respectively. Note that it is possible to have a final solution where the block k is traversed in empty subaisles (subaisle that do not contain SKUs in $\mathcal{L}$). We thus introduce for each empty subaisle (k,l), a virtual configuration $c \in \mathcal{C}^{all}_{k,l}$ with $q^c_s = 0, \forall s \in \mathcal{I}$ and $\tilde{d}^c_{k,l}$ equal to the distance associated with a traversal of subaisle (k,l). Figure 5.9 illustrates an inter-block VI for a 2-block warehouse. The warehouse includes three subaisles for each block. Block 1 is reached in subaisle $(1,1)$ with a configuration $c \in \hat{\mathcal{C}}_{0,0}$. Then, $c_1 \in \mathcal{C}^{all}_{0,1}$ and $c_2 \in \mathcal{C}^{all}_{0,2}$ must be selected for the subaisles $(0,0)$ and $(0,1)$ with an approximated distance highlighted by orange dashed lines.

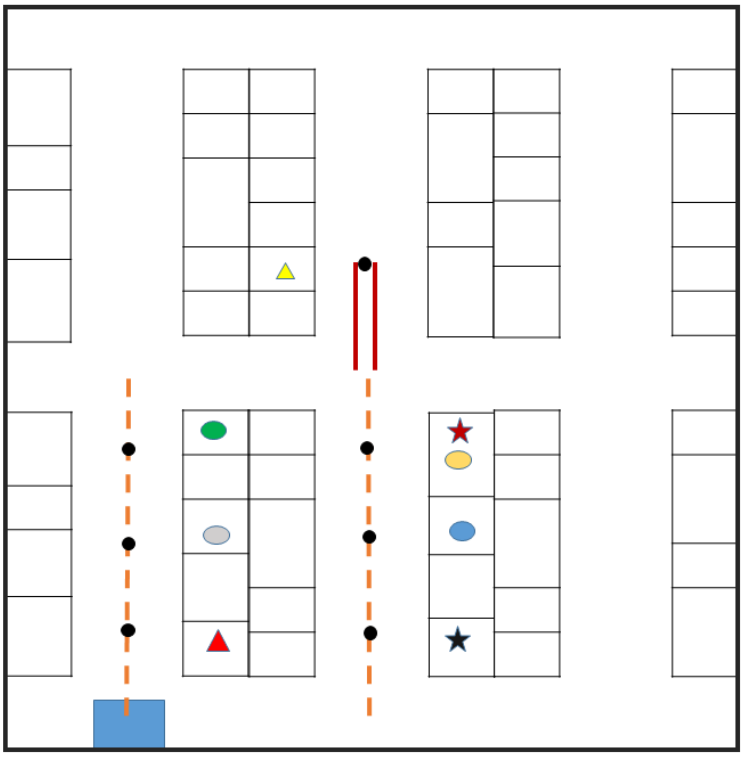


Figure 5.9 An illustrative example for inter-block VIs

To model the inter-block VIs, we use the following variables and constraints:

Variables:

— $k \in \{0, \cdots, K-2\}$, a binary variable β_k^1 that equals 1 if block $k+1$ has to be reached in its first subaisle, 0 otherwise.

— $k \in \{0, \cdots, K-2\}$, a binary variable β_k that equals 1 if block $k+1$ is reached in a subaisle $(k+1, l), l > 0$, 0 otherwise.

Constraints:

$$\beta_k^1 \geq \sum_{c \in \mathcal{C}_{k+1,0}} x_{k+1,0}^c \qquad \forall k \in \{0, \cdots, K-2\} \tag{5.13}$$

$$\beta_k \geq \sum_{c \in \mathcal{C}_{k+1,l}} x_{k+1,l}^c \qquad \forall k \in \{0, \cdots, K-2\}, \forall l \in \mathcal{L} \setminus \{0\} \tag{5.14}$$

$$\beta_k^1 \leq \sum_{l=0}^{L-1} \sum_{c \in \mathcal{C}_{k,l}^{all}} x_{k,l}^c \qquad \forall k \in \{0, \cdots, K-2\} \tag{5.15}$$

$$2 \cdot \beta_k \leq \sum_{l=0}^{L-1} \sum_{c \in \mathcal{C}_{k,l}^{all}} x_{k,l}^c \qquad \forall k \in \{0, \cdots, K-2\} \tag{5.16}$$

$$\beta_{k+1} \leq \beta_k \qquad \forall k \in \{0, \cdots, K-3\} \tag{5.17}$$

5.5 The SCH-MS+ formulation

In this section, we propose an adaptation of the formulation introduced in [10], referred to as SCH-PRP, to solve the PRP in mixed-shelves warehouses. For the sake of completeness, we first describe SCH-PRP, and then we discuss details on how to adapt it to our problem.

5.5.1 The SCH-PRP formulation

SCH-PRP works on a directed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A})$ that models the warehouse layout. The node set $\mathcal{V}'$ is defined as the intersection points between the aisles and the cross aisles (*i.e.*, top and bottom of subaisles). Each arc $a \in \mathcal{A}$ links two adjacent cross aisles or two adjacent aisles. Figure 5.10 depicts an example of a warehouse and its associated graph $\mathcal{G}'$. Note that since SCH-PRP solves the classical PRP, $\mathcal{G}'$ spans all the subaisles that contain picking locations to visit. The arcs associated with those subaisles $\mathcal{A}^m \subset \mathcal{A}$ are called "mandatory arcs". Each arc a is associated with a subaisle $(k-a, l-a)$. Furthermore, let $\mathcal{A}_l \subset \mathcal{A}$ be the set of cross aisle arcs located between aisles l and $l+1$.

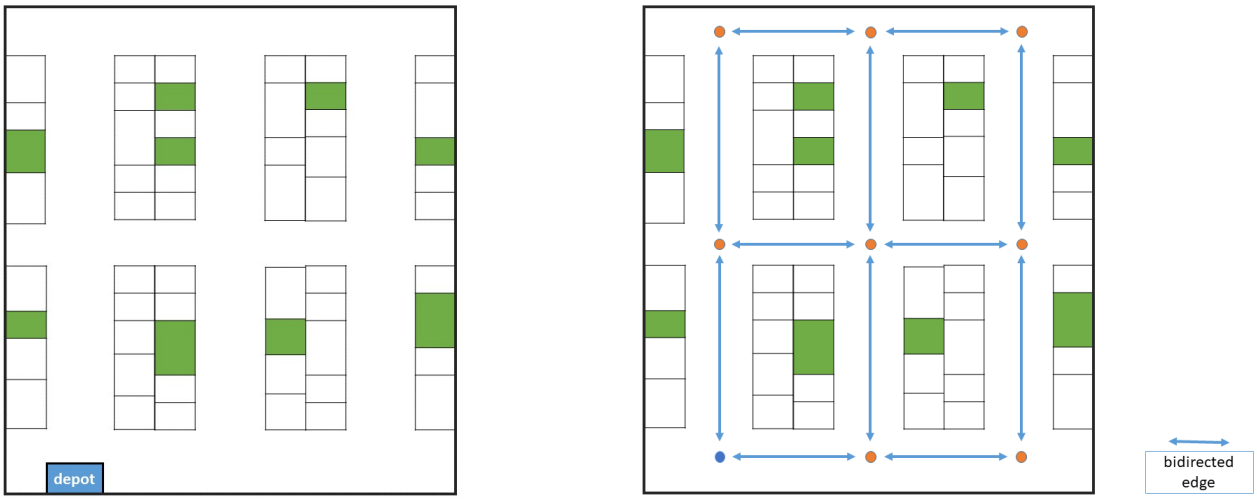


Figure 5.10 Warehouse layout and its associated graph representation

The fundamental idea behind the SCH-PRP formulation is to model the classical PRP as a Steiner rectilinear TSP in $\mathcal{G}'$ (with $\mathcal{A}^m$ being the set of mandatory arcs). In the solution, arcs in $\mathcal{A}_l$ are optional, that is, they can be traversed or not. Similarly, each arc spanning an empty subaisle (*i.e.*, a subaisle without picking locations to visit) can be traversed or not. Finally, each mandatory subaisle must be traversed using an arc that either traverses it or slopes it. The slopes can be performed using transitions (IV), (V), or (VI) (see Figure 5.4). Let $\bar{\mathcal{A}}^m = \{a \in \mathcal{A}^m : a = (b_{(k_a,l_a)}, t_{(k_a,l_a)})\}$ be the set of arcs on which the slopes are modeled and $\mathcal{T}_a^{'}$ the set of slopes associated with arc $a \in \bar{\mathcal{A}}^m$. A traversing cost c_a is associated with each arc $a \in \mathcal{A}$ and a sloping cost $c_{a,t}$ is associated with each arc $a \in \bar{\mathcal{A}}^m$ and each transition $t \in \mathcal{T}_a^{'}$. Finally, the sets, parameters, and variables used in the formulation are summarized in Table A.1.

Table 5.1 Notation used in the SCH-PRP formulation

Sets	
$\mathcal{A}$	set of all arcs
$\mathcal{A}^m$	set of all mandatory arcs
$\mathcal{A}^{d-}$	set of all arcs leaving the depot
$\mathcal{A}^{d+}$	set of all arcs entering the depot
$\mathcal{A}_l$	cross aisle arcs between aisles l and $l+1$
$\bar{\mathcal{A}}^m$	mandatory arc subset ($\bar{\mathcal{A}}^m = \{a \in \mathcal{A}^m : a = (b_{(k_a,l_a)}, t_{(k_a,l_a)})\}$)
$\mathcal{T}_a^{'}$	slopes transition set for arc a ($\mathcal{T}_{\cdot}^{'} = \{IV, V, VI\}$)
$\delta^-(a)$	cut set yielding all successor arcs of $a = (i,j)$
	(*i.e.*, $\delta^-(a) = \{b \in \mathcal{A} : b = (i', j')$ & $j = i'\}$)
$\delta^+(a)$	cut set yielding all predecessor arcs of $a = (i,j)$
	(*i.e.*, $\delta^-(a) = \{b \in \mathcal{A} : b = (i', j')$ & $i = j'\}$)
$\mathcal{B}^-$	set of all arcs that are successor arcs of arcs in $\mathcal{B}$ who end-vertex is not contained in the vertex set induced from all arcs in $\mathcal{B}$
Parameters	
c_a	cost of traversing arc a
$c_{a,t}$	cost of sloping arc a with transition t
a'	inverted arc of a
n	the index of the last aisle that contains at least an SKU to pick
(k_a, l_a)	subaisle associated with arc a
Variables	
$x_a \in \{0,1\}$	equals 1 if arc a is traversed, 0 otherwise.
$y_{a,t} \in \{0,1\}$	equals 1 if arc a is covered with slope t, 0 otherwise.
$z_{a,b} \in \{0,1\}$	equals 1 if arc b is traversed after arc a, 0 otherwise.
$o_a \in \{0,1\}$	equals 1 if a is the first arc of the tour, 0 otherwise.
$d_a \in \{0,1\}$	equals 1 if a is the last arc of the tour, 0 otherwise.

$$\min Z = \sum_{a \in \mathcal{A}} c_a x_a + \sum_{a \in \mathcal{A}^m} \sum_{t \in \mathcal{T}'_a} c_{a,t} y_{a,t} \tag{5.18}$$

$$x_b \geq \sum_{a \in \delta^+(b)} z_{a,b} \qquad \forall b \in \mathcal{A} \tag{5.19}$$

$$x_a \geq \sum_{b \in \delta^-(a)} z_{a,b} \qquad \forall a \in \mathcal{A} \tag{5.20}$$

$$\sum_{b \in \delta^-(a)} z_{a,b} \geq x_a \qquad \forall a \in \mathcal{A} \setminus \mathcal{A}^{d+} \tag{5.21}$$

$$\sum_{b \in \delta^-(a)} z_{a,b} + d_a \geq x_a \qquad \forall a \in \mathcal{A}^{d+} \tag{5.22}$$

$$\sum_{b \in \delta^+(a)} z_{a,b} \geq x_b \qquad \forall b \in \mathcal{A} \setminus \mathcal{A}^{d-} \tag{5.23}$$

$$\sum_{a \in \delta^+(b)} z_{a,b} + o_a \geq x_b \qquad \forall b \in \mathcal{A}^{d-} \tag{5.24}$$

$$\sum_{t \in \mathcal{T}'_a} y_{a,t} + x_a + x_{a'} \geq 1 \qquad \forall a \in \bar{\mathcal{A}}^m \tag{5.25}$$

$$\sum_{a \in \mathcal{A}^{d-}} o_a = 1 \tag{5.26}$$

$$x_a \geq o_a \qquad \forall a \in \mathcal{A}^{d-} \tag{5.27}$$

$$\sum_{a \in \mathcal{A}^{d+}} d_a = 1 \tag{5.28}$$

$$x_a \geq d_a \qquad \forall a \in \mathcal{A}^{d+} \tag{5.29}$$

$$\sum_{t \in \{IV,VI\}} y_{a,t} \leq \sum_{b \in \delta^+(a)} x_b \qquad \forall a \in \bar{\mathcal{A}}^m \tag{5.30}$$

$$\sum_{t \in \{V,VI\}} y_{a,t} \leq \sum_{b \in \delta^-(a)} x_b \qquad \forall a \in \bar{\mathcal{A}}^m \tag{5.31}$$

$$\sum_{a \in \mathcal{A}_l} x_a \geq 2 \qquad \forall l \in \{0, \cdots, n-1\} \tag{5.32}$$

$$\sum_{a \in \mathcal{B}} x_a \leq \sum_{a \in \mathcal{B}^-} x_a |\mathcal{A}| \qquad \forall \mathcal{B} \subseteq \mathcal{A} \setminus \{\mathcal{A}^{d-} \cup \mathcal{A}^{d+}\} \tag{5.33}$$

$$x_a, y_{a,t}, z_{a,b}, o_a, d_a \in \{0,1\} \qquad \forall a, b \in \mathcal{A} \tag{5.34}$$

The objective minimizes the total cost, which results from traversing or sloping arcs. Constraints (5.19) and (5.20) link x (arcs selection) and z (arcs sequencing) variables. Constraints (5.21) and (5.22) ensure that all but the last traversed arc has a successor. Similarly, constraints (5.23) and (5.24) enforce the existence of a predecessor for all but the first traversed arc. Constraints (5.25) ensure that every subaisle that must be visited is traversed

or sloped. Constraints (5.26) and (5.27) uniquely identify the first arc of a tour, while constraints (5.28) and (5.29) uniquely identify the last arc of a tour. Constraints (5.30) and (5.31) force each slope to be connected to a traversal and constraints (5.32) impose that the arcs associated with all cross aisles located between aisles l and $l+1$ are traversed at least twice. Constraints (5.33) ensure connectivity. Finally, constraints (5.34) define the domain of the variables. Since the number of constraints (5.33) grows exponentially with the size of $\mathcal{A}$, they are added in a lazy fashion as feasibility cuts in the solver.

5.5.2 Adaptation of the ILP

In the PRP-MS, we have to simultaneously select the picking locations from where to retrieve the SKUs and design the tour that visits the selected locations. Let $\mathcal{A}^{pm}$ be the set of pseudo-mandatory arcs. This set is made up of all the subaisles that store SKUs requested in $\mathcal{I}$. It is worth noting that these subaisles are not all necessarily visited in the optimal solution.

The set of transitions $\mathcal{T}_a^{pm}$ associated with each pseudo-mandatory arc depends on the picking locations selected in (k_a, l_a). To reduce its cardinality, we replicate the same rationale used to construct $\mathcal{C}_{k,l}$. Each transition $t \in \mathcal{T}_a^{pm}$ is mapped to a configuration $c \in \mathcal{C}_{k,l}$, except for the transition that contains all SKUs in (k, l), that is, $c \in \mathcal{C}_{k,l}^{all}$. The latter is triplicated into three transitions: top slope, bottom slope, double slope.

We adapt the formulation (5.18)-(5.34) by introducing the following modifications:

— We define $\mathcal{A}^{pm}$ as the set of all pseudo-mandatory arcs
— We replace $\mathcal{T}'_a$ by $\mathcal{T}_a^{pm}$ and $\bar{\mathcal{A}}^m$ by $\bar{\mathcal{A}}^{pm}$
— We define for each arc $a \in \bar{\mathcal{A}}^{pm}$ and each SKUs $s \in \mathcal{I}$, q_a^s as the quantity of SKU s picked when arc a is traversed
— We define for each arc $a \in \bar{\mathcal{A}}^{pm}$, transition $t \in \mathcal{T}_a^{pm}$, and SKUs $s \in \mathcal{I}$, $q_{a,t}^s$ as the quantity of SKUs s picked when arc a is covered using transition t
— We define for each arc $a \in \bar{\mathcal{A}}^{pm}$ and transition $t \in \mathcal{T}_a^{pm}$, $c_{a,t}$ as the distance traveled when sloping arc a using transition t
— We define $\hat{\mathcal{T}}_a^{pm} \subset \mathcal{T}_a^{pm}$ as the set of transitions where the picker exits subaisle (k_a, l_a) from the bottom
— We define $\check{\mathcal{T}}_a^{pm} \subset \mathcal{T}_a^{pm}$, the set of transitions where the picker exits subaisle (k_a, l_a) from the top
— We define $\check{\hat{\mathcal{T}}}_a^{pm} \subset \mathcal{T}_a^{pm}$, the set of transitions where the picker exits subaisle (k_a, l_a)

from both sides

— We replace constraint (5.25) by (5.35)

$$\sum_{a\in\bar{\mathcal{A}}^{pm}}\left(\sum_{t\in\mathcal{T}_a^{pm}} q_{t,a}^s\cdot y_{a,t}+q_a^s\cdot(x_a+x_{a'})\right)\geq r_s \qquad \forall s\in\mathcal{I} \tag{5.35}$$

— We replace constraint (5.30) by (5.36)

$$\sum_{t\in\check{\mathcal{T}}_a^{pm}\cup\hat{\check{\mathcal{T}}}_a^{pm}} y_{a,t}\leq\sum_{b\in\delta^+(a)} x_b \qquad \forall a\in\bar{\mathcal{A}}^{pm} \tag{5.36}$$

— We replace constraint (5.31) by (5.37)

$$\sum_{t\in\hat{\mathcal{T}}_a^{pm}\cup\hat{\check{\mathcal{T}}}_a^{pm}} y_{a,t}\leq\sum_{b\in\delta^-(a)} x_b \qquad \forall a\in\bar{\mathcal{A}}^{pm} \tag{5.37}$$

— We remove constraint (5.32), because it is redundant

5.5.3 Adaptation of the VIs

The intra-block and the inter-block VIs are reformulated as follows:

$$2\cdot\sum_{t\in\overline{\mathcal{T}}_a^{pm}} y_{a,t}\leq\sum_{b\in\mathcal{A}^m:k_b=k_a,l_b\neq l_a}(x_b+x_{b'}) \qquad \forall a\in\bar{\mathcal{A}}^{pm} \tag{5.38}$$

$$\beta_k\geq\sum_{t\in\mathcal{T}_a^{pm}} y_{a,t}+x_a+x_{a'} \qquad \forall k\in\mathcal{K}\setminus\{K-1\},\ \forall a\in\bar{\mathcal{A}}^{pm}:k_a=k+1 \tag{5.39}$$

$$2\cdot\beta_k\leq\sum_{a\in\bar{\mathcal{A}}^{pm}:k_a=k+1}(x_a+x_{a'}) \tag{5.40}$$

5.6 Computational experiments

We performed two sets of experiments to assess and analyze the performance of our LBBD method. The first set (Section 6.1) was carried on 1-block warehouse instances. These experiments aim to: 1) measure the impact of the LBF and the VIs in the effectiveness and efficiency of our method; and 2) study the behaviour of our method and the model introduced in section 5 (referred to as `SCH-MS+`) in 1-block warehouses. The second set (Section 6.2) establishes a comparison between our LBBD procedure and `SCH-MS+` in multi-block warehouses. For simplicity, in the reminder of this section we refer to the standard version of our algorithm as `LBBD` and to its enhanced version as `LBBD+`.

All computations were performed on a 64-bit laptop equipped with an Intel core i5-6300U CPU (2.40 GHz), 16 gigabytes of main memory, running on Windows 10. All methods were implemented in C++ (visual studio 2019), using IBM ILOG CPLEX Optimizer 12.8 to solve the ILPs. Finally, all methods were given a time limit of $TL = 300s$.

5.6.1 Results in 1-block warehouses

We ran `LBBD`, `LBBD+` and `SCH-MS+` on the `Set-1B` testbed introduced in [35]. For the sake of completeness, we briefly describe the data here. The set contains 108 instances. All instances are defined on a 1-block warehouse layout. The instances have four varying parameters, namely: the number of aisles L, the number of picking locations per aisle R, the cardinality of picklist $|\mathcal{I}|$, and a parameter γ that controls the duplication of SKUs in the warehouse. Table 5.2 summarizes the values adopted for each parameter. There is one instance for each combination of parameter values (*i.e.*, $3 \times 3 \times 3 \times 4 = 108$). The SKUs are assigned to shelves as follows. First, the total number of SKUs stored in the warehouse, denoted as ξ, is fixed as: $\xi = max(|\mathcal{I}|, \lceil L \cdot R/\gamma \rceil)$. Then, the SKUs are cast into categories using an ABC classification. The A class includes 20% of all SKUs, the B class 30%, and the C class the remaining 50%. One unit of each SKU is first assigned to a randomly selected picking location to ensure the availability of each SKU in the warehouse. The shelves associated with the remaining picking locations receive SKUs from the A, B, and C classes with 80%, 15%, and 5% probability, respectively. Once a class is selected, an SKU s is randomly selected from the corresponding class. Then, the quantity of s is uniformly drawn from the set $\{1, 2, 3\}$.

The picklist $\mathcal{I}$ is filled by randomly selecting SKUs from the A, B, and C classes with a probability of 80%, 15%, and 5%, respectively. Finally, the number of requested units for each SKU s in the picklist is randomly drawn as an integer from the interval $[1, min(6, \bar{q}_s)]$, where $\bar{q}_s$ is the total supply of SKU s in the warehouse. For further details about the data set generation, we refer the reader to [35].

Table 5.2 Warehouse factors for `Set-1B`

Parameters	Values
L	5, 25, 100
R	30, 60, 180
$\|\mathcal{I}\|$	3, 7, 15, 30
γ	5, 10, 40

In these experiments, the subproblem in both LBBD and LBBD+ is solved using the efficient ILP of [35], which is specially designed for 1-block warehouses. Table 5.3 reports the results grouped by number of aisles. For each method, the table reports the following metrics:

— #opt: the number of instances solved to optimality for each value of L. An instance is deemed solved to optimality if $(UB_{\cdot} - LB_{\cdot}) < \epsilon$, where $\epsilon = 1$.

— gap: the average gap associated with each parameter value. For each instance, the gap is computed as: $\frac{UB_{\cdot} - z^*_{\cdot}}{z^*_{\cdot}}$, where $z^*_{\cdot}$ is the objective function of the optimal solution returned by the ILP introduced in [35]. As mentioned earlier, this ILP is currently the state-of-the-art method to solve the PRP in 1-block warehouses but it cannot be exploited in the more general multi-block setting. Note that SCH-MS+ fails to find a feasible solution in 15 out of the 108 instances (1 for $L = 25$ and 14 for $L = 100$). We thus distinguish two metrics for the gap: gap^1 which is computed by taking into account all the instances for each parameter value and gap^2 which is computed by only taking into account the instances for which SCH-MS+ returns a feasible solution for each parameter value.

Table 5.3 Comparison between the performance of LBBD+ and SCH-MS+ for Set-1B

	LBBD		LBBD+			SCH-MS+	
L	#opt	gap^1	#opt	gap^1	gap^2	#opt	gap^2
5	(1/36)	73 %	(35/36)	0.00%	0.00%	(36/36)	0.00%
25	(0/36)	322 %	(28/36)	1.25%	1.13%	(11/36)	17.48%
100	(0/36)	559 %	(16/36)	1.90%	1.01 %	(1/36)	552.56%
all	(1/108)	318 %	(79/108)	1.05%	0.66%	(48/108)	137.29%

We first compare the performance of LBBD and LBBD+. We observe that LBBD only finds the optimal solution for 1 out of the 108 instances. Furthermore, LBBD reported solutions with an average gap with respect to the optimal solutions ranging from 73% (for $L = 5$) to 559% (for $L = 100$). On the other hand, LBBD+ finds the optimal solution to nearly three quarters of the instances (79/108). A closer look at the average gap, reveals that in the remaining cases, the method delivers near-optimal solutions. Indeed, in the worst case scenario (i.e., instances with 100 aisles), the average gap is less than 2%. Moreover, in 4 out the 29 instances without proven optimum, the method finds the optimal solution but does not prove optimality. We conclude that the LBF and the VIs have a notorious and positive impact on the effectiveness of our LBBD-based method.

When comparing the performance of LBBD+ and SCH-MS+, we observe that on instances with $L = 5$, both methods are competitive with a slight advantage for SCH-MS+ (36 proven optima vs. 35). However, increasing the number of aisles yields to a completely different behavior. Indeed, running on instances with L=25 and L=100, LBBD+ finds a larger number of proven optima (28 and 16 vs. 11 and 1) and delivers solutions with significantly smaller gaps (1.13% and 1.01% vs. 17.48% and 552.56%).

In light of these results, we conclude that our LBBD-based method is capable of delivering, in less than 5 minutes, excellent-quality solutions in 1-block warehouses. This results are remarkable taking into account that the method was designed for a more general problem and therefore (contrary to the state-of-the-art method) it does not exploit the well known mathematical properties of the 1-block problem setting.

5.6.2 Results in multi-block warehouses

To the best of our knowledge there exists no benchmark results for the multi-block PRP-MS. Therefore, we generated a new data set, named Set-MB following the same procedure employed in [35] to generate the Set-1B testbed. Table 5.4 summarizes our instance parameters. Similarly to the Set-1B set, we generated one instance per combination of parameter values, that is 96 instances (*i.e.*, $2 \times 4 \times 2 \times 3 \times 2 = 96$).

Table 5.4 Warehouse factors for Set-MB

Parameters	Values
K	2, 3
L	5, 15, 20, 30
R	30, 60
$\|\mathcal{I}\|$	7, 15, 30
γ	10, 40

The subproblem at each iteration of LBBD+ is solved using SCH-PRP. Tables 5.5 and 5.6 report the main results for instances with two and three blocks, respectively. In particular, the tables report the following metrics:

— t_L (resp. t_S) the CPU time (in seconds) associated with LBBD+ (resp. SCH-MS+). TL means that the method reached the time limit without closing the gap.

— Δ the gap between the solutions returned by LBBD+ and SCH-MS+ computed as: $\frac{z^S - UB}{UB}$, where z^S is the solution returned by SCH-MS+. Hence, a positive value of Δ indicates

that LBBD+ outperforms SCH-MS+ while a negative value of Δ represents the opposite.

In the following we refer to each instance with the following tuple: (K,L,R,$|\mathcal{I}|$, γ).

Table 5.5 Comparison between the performance of LBBD+ and SCH-MS+ for 2-block instances

	$\|\mathcal{I}\| = 7$			$\|\mathcal{I}\| = 15$			$\|\mathcal{I}\| = 30$		
(L,R)	t_L	t_S	Δ	t_L	t_S	Δ	t_L	t_S	Δ
$\gamma = 10$									
(5,30)	0.158	0.374	0%	0.39	0.449	0%	0.727	0.8	0%
(5,60)	0.144	0.219	0%	4.913	16.043	0%	TL	2.663	0%
(15,30)	5.601	35.995	0%	0.477	44.725	0%	TL	62.109	0%
(15,60)	0.204	TL	1%	7.78	12.736	0%	41.229	119.055	0%
(20,30)	3.024	98.334	0%	TL	TL	0%	0.965	TL	14%
(20,60)	0.254	221.998	0%	0.514	TL	0%	TL	TL	8%
(30,30)	250.363	TL	24%	51.37	TL	62%	TL	TL	79%
(30,60)	2.837	TL	149%	280.725	TL	101%	TL	TL	294 %
$\gamma = 40$									
(5,30)	0.734	1.397	0%	0.59	0.498	0%	TL	0.648	0%
(5,60)	2.812	5.03	0%	13.411	4.36	0%	TL	8.189	0%
(15,30)	0.673	5.314	0%	10.703	TL	8%	TL	TL	3%
(15,60)	0.427	89.504	0%	5.083	TL	6%	TL	TL	4%
(20,30)	1.026	TL	5%	1.723	TL	23%	TL	TL	-2%
(20,60)	10.602	100.685	0%	4.909	35.543	0%	TL	TL	63%
(30,30)	0.295	1.897	0%	1.392	99.473	0%	TL	TL	74%
(30,60)	0.561	185.502	0%	2.725	TL	135%	17.737	TL	291 %

We observe from Tables 5.5 and 5.6 that both LBBD+ and SCH-MS+ are competitive on instances with five aisles (*i.e.*, $L = 5$). Indeed, SCH-MS+ finds optimal solutions for all instances while LBBD+ comes short only on one instance $(3, 5, 30, 30, 40)$. In 16/24 instances, LBBD+ was faster to find the optimal solution than SCH-MS+, while for the remaining instances SCH-MS+ exhibited a better computational performance.

The balance starts tipping towards LBBD+ on instances with $L = 15$. LBBD+ closes the optimality gap in 16/24 vs. 10/24 for SCH-MS+. In 3/24 instances, SCH-MS+ finds better solutions than LBBD+, reporting gaps of -2%, -4%, and -9, while LBBD+ dominates SCH-MS+ in 9/24 instances, with a maximum gap of 103%.

Table 5.6 Comparison between the performance of LBBD+ and SCH-MS+ for 3-block instances

	$\|\mathcal{I}\| = 7$			$\|\mathcal{I}\| = 15$			$\|\mathcal{I}\| = 30$		
(L,R)	t_L	t_S	Δ	t_L	t_S	Δ	t_L	t_S	Δ
$\gamma = 10$									
(5,30)	0.17	1.946	0%	1.148	1.215	0%	TL	6.058	-3%
(5,60)	0.097	0.641	0%	0.665	2.282	0%	288.94	3.923	0%
(15,30)	0.434	71.51	0%	TL	TL	-2%	TL	TL	-9%
(15,60)	3.794	TL	0%	6.261	162.315	0%	TL	148.419	-4%
(20,30)	TL	TL	-4%	TL	TL	42%	TL	TL	1%
(20,60)	TL	TL	5%	TL	TL	61%	2.671	TL	6%
(30,30)	TL	TL	0%	TL	TL	81%	TL	TL	64%
(30,60)	TL	TL	234%	TL	TL	300%	TL	TL	77%
$\gamma = 40$									
(5,30)	0.497	1.769	0%	0.863	1.387	0%	TL	6.202	0%
(5,60)	2.863	4.504	0%	5.313	139.312	0%	TL	82.368	0%
(15,30)	0.459	TL	0%	1.235	TL	21%	TL	TL	6%
(15,60)	0.65	7.701	0%	4.523	TL	13%	TL	TL	103%
(20,30)	3.285	TL	0%	24.997	TL	48%	TL	TL	71%
(20,60)	0.965	TL	14%	15.516	TL	96%	TL	TL	122%
(30,30)	14.598	TL	0%	2.248	TL	127 %	TL	TL	196%
(30,60)	1.612	TL	70%	1.761	TL	120%	TL	TL	429 %

On instances with $L > 15$, LBBD+ performs much better than SCH-MS+ (except for two instances). Indeed, LBBD+ closes the optimality gap in 26/48 instances vs. only 7/48 instances for SCH-MS+. Note that on 3-block instances, SCH-MS+ could not find any proven optimal solution while LBBD+ finds provably optimal solution for nine instances. Furthermore, LBBD+ returns better solutions than SCH-MS+ in 34/48, with a remarkable maximum gap of 429%.

In conclusion, both algorithms perform well on small warehouses with $L = 5$. However, LBBD+ clearly dominates SCH-MS+ for medium and large-sized warehouses (*i.e.*, warehouses with $L >= 15$).

5.7 Conclusion

In this paper, we study the picker routing problem in mixed-shelves warehouses. Unlike most existing research, we assume in our problem definition that the warehouse may have a multi-block layout.

To solve the problem, we propose a logic-based Benders decomposition procedure that selects the picking locations from where to retrieve the SKUs in the master problem and designs the picking tour that visits the selected locations in the subproblem. To speed-up our LBBD, various algorithmic enhancements were implemented including a LBF and two sets of VIs. We also proposed an adaptation of the state-of-the-art ILP for the classical PRP (`SCH-PRP`) to work on mixed-shelves warehouses.

In the computational experiments, we first compared the standard version of our LBBD to its enhanced version (`LBBD+`). Results showed that the LBF and the VIs have a strong impact on the performance of our method. Furthermore, we compared `LBBD+` to `SCH-MS+` in 1-block and multi-block warehouses configurations. We observed that both methods are competitive for small warehouses ($L = 5$) while `LBBD+` systematically outperforms `SCH-MS+` on medium and large warehouses ($L >= 15$).

In our problem description, we assume a single order picker that has been assigned a picking list constructed in an upstream level. In practice, the assignment of picklists to pickers plays a role in the performance of the picking process. Further research should focus on extending our model to an integrated version that considers multiple order pickers and multiple orders to batch into picking lists before designing the tours that retrieve each picklist.

CHAPITRE 6 STOCKAGE TRADITIONNEL VERSUS STOCKAGE À ÉTAGÈRES MIXTES DANS DES ENTREPÔTS DÉCOUPÉS EN ZONES

6.1 Introduction

Avec le développement des technologies de communication et d'information (TIC), les activités liées au commerce électronique (e-commerce) ont connu une croissance considérable dans les dernières décennies. Au Canada, par exemple, le revenu total des détaillants d'e-commerce est passé de 21.9 milliards de dollars en 2017 à 25.3 milliards de dollars en 2019, entrainant ainsi une augmentation de 15.5% dans cette période [4]. Cette croissance a été amplifiée par l'escalade de la COVID-19. En effet, les différentes restrictions sociales imposées par plusieurs pays à travers le monde pour limiter la propagation du virus ont poussé les consommateurs à se tourner vers les plateformes électroniques pour faire leurs achats [90].

Ce nouvel environnement a poussé les entreprises à diversifier leur catalogue de produits afin d'attirer une large clientèle. De plus, ces entreprises doivent adapter les différentes activités de la chaîne logistique, et plus particulièrement les activités d'entreposage, pour répondre à un nouveau pattern de demande caractérisé par un grand nombre de petites commandes à préparer dans des délais de livraison très courts. Cette adaptation passe par l'implémentation d'un bon système de gestion des activités de stockage et de préparation des commandes.

Les entrepôts d'e-commerce adoptent généralement l'un des deux systèmes suivants : *article vers humain* et *humain vers article*. Dans les systèmes *article vers humain*, une flotte de robots se déplace dans l'entrepôt et apporte des étagères mobiles à des opérateurs humains stationnaires qui récupèrent les items, appelés aussi unités de gestion des stocks (UGS), appartenant aux commandes des clients [66]. Les entrepôts implémentant ces systèmes, appelés *entrepôts robotisés*, permettent une gestion efficace de la préparation des commandes. Cependant, ces systèmes induisent des coûts d'investissement très élevés, souvent insupportables pour les petites et moyennes entreprises. De plus, ils souffrent d'un manque de flexibilité qui les rend inappropriés pour les entrepôts confrontés à une charge de travail variable causée par une demande volatile (les ventes saisonnières ou le Vendredi Fou, par exemple) [10].

De l'autre côté, les systèmes *humain vers article* sont les plus populaires en pratique, car ils nécessitent des investissements moindres et offrent une plus grande flexibilité pour faire face aux périodes de pics [13]. Dans ces systèmes, un ensemble de préparateurs de commandes

se déplacent dans l'entrepôt, s'arrêtent dans des *emplacements de collecte* pour récupérer des items stockés dans des étagères fixes et les amènent à l'unité d'emballage. Les entrepôts implémentant ces systèmes sont généralement structurés en *blocs*, composés d'*allées de ramassage* et de *couloirs transversaux*. Les allées de ramassage permettent aux préparateurs d'accéder aux étagères et de récupérer les items. Les couloirs transversaux divisent les allées en *sous-allées* et la zone d'entreposage en blocs. À partir d'une allée transversale, un préparateur de commandes peut entrer ou sortir d'une sous-allée, ou se déplacer vers d'autres allées. Un exemple d'un entrepôt disposé en deux blocs est donné dans la figure 6.1.

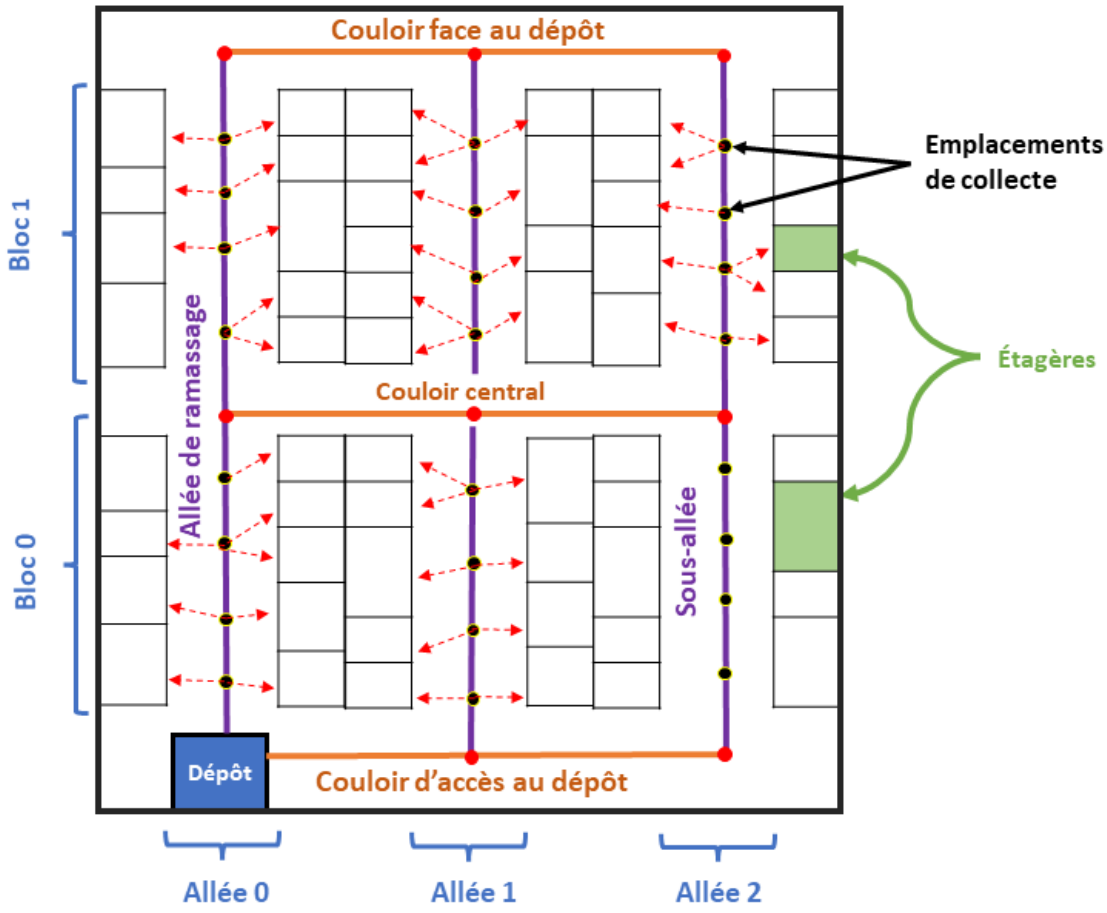


FIGURE 6.1 Représentation d'un entrepôt disposé en deux blocs

Les entrepôts traditionnels de type humain vers article implémentent généralement une politique de stockage dite *dédiée* dans laquelle chaque item est stocké dans une seule étagère ou dans des étagères collées les unes aux autres. De plus, chaque préparateur de commandes opère à partir d'un dépôt central et traite une commande à la fois (collecte par commandes)

ou un ensemble de commandes (collecte par lots) au cours de chaque tournée de collecte. Dans les deux cas, il est guidé par une liste dans laquelle figurent les items à collecter. Une grande partie de la littérature scientifique s'intéresse à l'optimisation du processus de préparation des commandes dans un tel contexte. Le problème le plus étudié est le problème de routage d'un préparateur (PRP). Il cherche à déterminer la tournée optimale d'un seul préparateur de commandes, en termes de distance à parcourir, pour récupérer les items inclus dans sa liste de collecte à partir de leurs emplacements respectifs. Bien que ce problème puisse être modélisé comme un problème de voyageur de commerce (TSP) [32], de nombreux chercheurs ont développé des méthodes "ad hoc" qui modélisent le problème comme un Steiner-TSP et exploitent la disposition en bloc des entrepôts. Ratliff et Rosenthal [27] ont présenté un travail fondamental qui limite le nombre de possibilités de traverser une sous-allée de ramassage dans une solution optimale à six, et ce indépendamment du nombre d'emplacements de collecte à visiter dans cette allée. De plus, ils ont proposé un programme dynamique pour résoudre le problème pour les entrepôts disposés en un bloc. Roodbergen et De Koster [11] ont étendu ces travaux aux entrepôts disposés en deux blocs. Pansart et al. [29] et Schiffer et al. [10] ont proposé des formulations de programmation dynamique et de programmation linéaire en nombres entiers (PLNE) pour les entrepôts multi-blocs. Enfin, Masae et al. [72, 91] ont adapté le programme dynamique introduit dans [27] à des catégories d'entrepôts disposés de manière non conventionnelle.

Comme mentionné dans [66], les entrepôts traditionnels sont généralement inadaptés pour faire face à la charge de travail causée par le pattern de demande associé au commerce électronique. Au lieu d'investir dans des systèmes robotisés, de nombreux entrepôts s'adaptent à ce nouveau pattern en apportant de nouvelles stratégies au niveau de la politique de stockage et de l'organisation de la préparation des commandes. Dans ce chapitre, on s'intéressera principalement aux stratégies suivantes : le stockage à étagères mixtes et le découpage de l'espace de collecte en zones.

Dans un entrepôt adoptant une politique de stockage à étagères mixtes, le stock de chaque item est divisé en petites unités qui sont dispersées dans différentes étagères un peu partout dans l'entrepôt. Cette politique est mise en place dans de nombreux centres de distribution modernes (Amazon Europe et Zalando, par exemple). Elle permet entre autres de réduire en moyenne la distance des tournées de collecte des préparateurs et d'optimiser l'utilisation de l'espace de stockage [5]. La littérature scientifique qui considère le stockage à étagères mixtes est relativement rare. Dans leur étude [17], Weidinger et Boysen s'intéressent à la meilleure manière de distribuer les items dans les étagères de l'entrepôt. Pour cela, ils définissent un

ensemble de repères correspondant à des entrepôts intermédiaires et formulent le problème d'affectation des items aux étagères avec l'objectif de minimiser la "dispersion" des items. Cette dernière est définie comme la distance maximale entre les UGS et les repères. Dans la partie expérimentale, ils proposent des perspectives managériales sur la fréquence à laquelle le réapprovisionnement des étagères devrait être effectué. Weidinger [34] et Goeke et al. [35] généralisent le PRP pour les entrepôts à étagères mixtes (PRP-MS). Étant donné un ensemble d'items à récupérer, le problème consiste à sélectionner les emplacements à partir d'où récupérer les items et de définir la plus courte tournée qui visite les emplacements sélectionnés. Dans [34], l'auteur propose une heuristique en deux étapes qui sélectionne d'abord les emplacements à partir d'où prélever les items en utilisant une règle de sélection, puis applique le programme dynamique introduit dans [27] pour trouver la tournée de collecte. Dans la partie expérimentale, il propose un ensemble de suggestions sur la pertinence d'adopter une politique de stockage à étagères mixtes en fonction de l'hétérogénéité de la liste de collecte. Goeke et al. [35] proposent une formulation PLNE compacte pour résoudre la PRP classique pour les entrepôts à un bloc seulement. La formulation est ensuite adaptée à plusieurs configurations d'entrepôt modernes, y compris le stockage à étagères mixtes. La principale contribution de ce travail est d'introduire un nouveau type de contraintes pour l'élimination des sous-tours en exploitant les caractéristiques d'une solution optimale pour les entrepôts à un bloc. Cependant, cette formulation est uniquement applicable pour les entrepôts à un bloc.

Le découpage de l'entrepôt en zones, appelé aussi *zonage*, désigne une stratégie dans laquelle l'espace de collecte est partitionné en zones. Chaque zone est définie par un ensemble d'allées où un ou plusieurs préparateurs récupèrent les items des commandes localisés dans cette zone. Cette stratégie permet de réduire le chevauchement des préparateurs puisque ces derniers sont affectés à des zones disjointes. De plus, chaque préparateur opère dans une zone plus petite, ce qui réduit le temps de déplacement et permet aux préparateurs de se familiariser avec l'emplacement des items [23]. La littérature sur la préparation des commandes dans les entrepôts décomposés en zones identifie deux approches pour la préparation des commandes (voir [9]) : le zonage *séquentiel* et le zonage *synchronisé*. Dans le zonage séquentiel, les zones de collecte sont classées selon un ordre (de la plus proche de l'unité d'emballage à la plus lointaine par exemple), et chaque liste de collecte est assemblée de manière progressive selon l'ordre préétabli. Chaque préparateur récupère la partie de la liste de collecte située dans sa zone et la dépose dans un bac qui est transféré à un préparateur de la zone suivante. Le processus est répété jusqu'à ce que toute la liste soit traitée. Afin d'optimiser le processus de préparation des commandes dans les entrepôts adoptant un zonage séquentiel, les travaux

de la littérature appliquent généralement des simulations et des modèles de files d'attente pour étudier l'impact de la variation du nombre de zones et de la taille du système [24] ou de la combinaison du zonage avec différentes stratégies de stockage et de préparation de commandes [25] sur la performance du système, c.-à-d. la date de fin de collecte des items.

Dans des entrepôts implémentant un zonage synchronisé, les préparateurs de toutes les zones opèrent simultanément sur une même commande (ou une vague de commandes). Chaque préparateur récupère la partie de la commande présente dans sa zone et la dépose dans un dépôt intermédiaire qui donne accès à un système de convoyage. Bien que ces systèmes nécessitent généralement un effort supplémentaire pour le processus de consolidation des items en commandes, plusieurs entrepôts modernes l'implémentent, car ils disposent de systèmes logistiques innovants, capables de gérer efficacement les opérations de consolidation des commandes [92]. Le critère considéré dans le zonage synchronisé diffère de celui considéré dans le zonage séquentiel. En effet, les entrepôts implémentant un zonage synchronisé organisent la préparation des commandes par vagues de commandes. Chaque vague courante contient des commandes urgentes qui doivent être préparées ensemble le plus rapidement possible. Minimiser la date de fin de collecte d'une vague de commandes dans ce cas revient à minimiser la distance de la plus grande tournée parmi les préparateurs opérant durant cette vague dans les différentes zones. La littérature qui étudie le processus de préparation de commandes dans un tel contexte est relativement limitée. Elle se concentre sur des décisions tactiques telles que la recherche du nombre optimal de zones dans un système de type *pick-and-sort* [7] ou l'étude de l'impact de la combinaison d'un zonage synchronisé avec différentes stratégies de stockage sur les performances du système [26]. Un autre concept rarement étudié dans la littérature est le zonage dynamique. Comme son nom l'indique, dans ce système, la taille de chaque zone peut être modifiée en temps réel. Saylam et al. [23] étudient un problème de préparation de commandes dans un entrepôt implémentant un zonage synchronisé et dynamique. Ils montrent dans la partie expérimentale que le passage d'une zone à deux zones diminue considérablement le délai d'exécution, mais l'impact se réduit de manière exponentielle avec l'augmentation du nombre de zones.

Dans ce chapitre, nous nous plaçons dans un environnement dans lequel les commandes arrivent en temps réel et le processus de préparation des commandes est effectué par vagues de commandes. Nous étudions la combinaison de deux pratiques émergentes dans les entrepôts de commerce électronique : la politique de stockage à étagères mixtes et le zonage synchronisé. Cette combinaison a été identifiée par la récente revue de littérature de Boysen et al. [66] comme une perspective de recherche intéressante et n'a, à notre connaissance, pas encore

été étudiée. Nous proposons deux méthodes exactes qui trouvent les tournées optimales pour traiter une vague de commandes, c'est-à-dire les tournées qui minimisent la distance de la plus longue tournée. La première méthode est une décomposition de Benders de type "logic-based" (LBBD), adaptée de [92], capable de résoudre le problème pour un nombre arbitraire de blocs. La deuxième méthode est une formulation PLNE compacte, adaptée de [35], qui est spécialisée pour les entrepôts disposés en un bloc. En outre, nous étudions l'impact du passage d'une politique de stockage dédiée à une politique de stockage à étagères mixtes dans les entrepôts appliquant un zonage synchronisé et démontrons que des améliorations significatives du processus de préparation des commandes peuvent être obtenues en passant à un stockage à étagères mixtes.

Ce chapitre est organisé de la manière suivante. Dans la section 2, nous décrivons formellement le problème. Dans la section 3, nous présentons la procédure LBBD. Dans la section 4, nous présentons la formulation PLNE compacte. Dans la section 5, nous présentons les résultats expérimentaux. Finalement, dans la section 6, nous apportons des conclusions et des perspectives de recherche.

6.2 Description du problème

On s'intéresse à un problème de routage d'un ensemble de préparateurs de commandes dans un entrepôt de e-commerce caractérisé par :

- **Un système de type *humain vers article* avec un zonage synchronisé**. Les allées de chaque zone sont déterminées en amont (zonage statique), avec un préparateur de commandes opérant dans chaque zone. De plus, chaque zone contient au coin inférieur gauche un dépôt donnant accès à un système de convoyage. Ainsi, chaque préparateur commence sa tournée à partir du dépôt associé à sa zone, se déplace dans les allées de sa zone, s'arrête dans des emplacements de collecte pour récupérer les items qui lui sont assignés et retourne à ce même dépôt. Dans ce qui suit, on utilisera l'indice z pour référer à un préparateur de commande ou à la zone dans laquelle il opère. Dans chaque zone, on retrouve deux couloirs transversaux aux extrémités des allées en plus de couloirs centraux qui divisent l'entrepôt en blocs.
- **Une politique de stockage à étagères mixtes**, à travers laquelle le stock associé à chaque item est décomposé en petites unités qui sont réparties de manière aléatoire dans les différentes étagères de l'entrepôt. Ainsi, chaque item peut se retrouver dans plusieurs étagères en quantités limitées et chaque étagère peut stocker différents items.

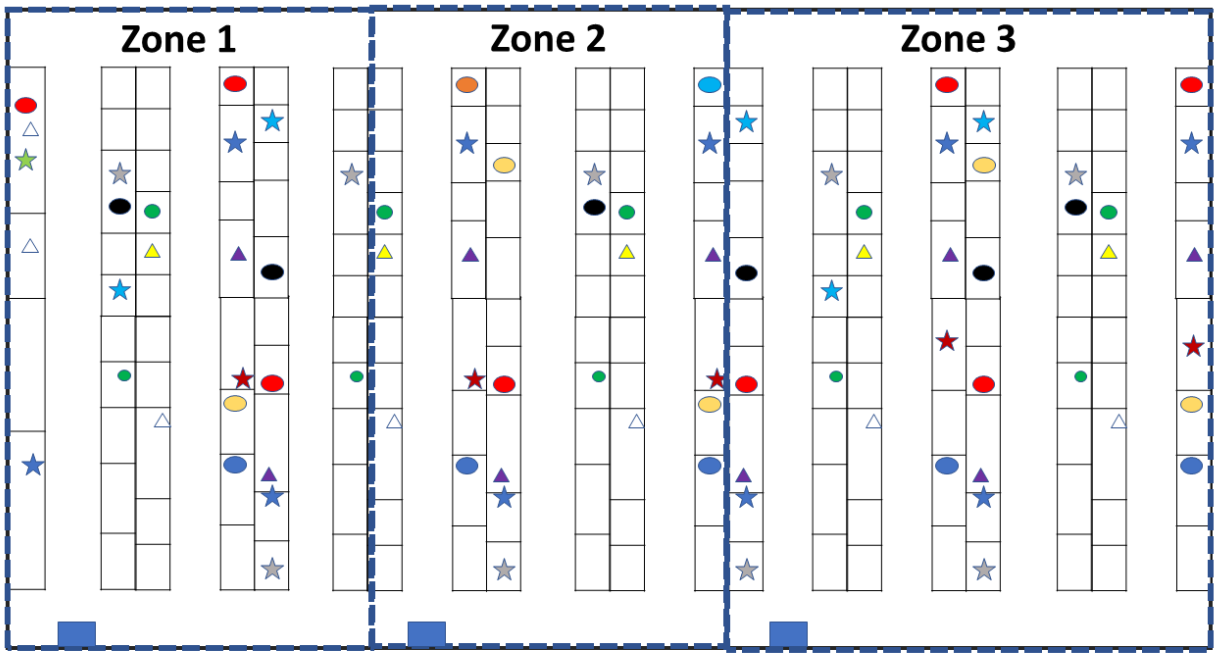


FIGURE 6.2 Représentation d'un entrepôt décomposé en zones appliquant une politique de stockage à étagères mixtes

Un exemple d'un entrepôt à étagères mixtes avec trois zones est représenté dans la figure 6.2. Formellement, soit $\mathcal{L} = \{0, \cdots, L-1\}$ l'ensemble des allées de collecte et $\mathcal{K} = \{0, \cdots, K-1\}$ l'ensemble des blocs de l'entrepôt. La sous-allée décrite par la paire (k, l) correspond à la partie de l'allée l située dans le bloc k. De plus, on définit $t_{(k,l)}$ (resp. $b_{(k,l)}$) comme l'extrémité supérieure (resp. l'extrémité inférieure) de la sous-allée (k, l). Dans ce qui suit, on considère que toutes les sous-allées (resp. tous les couloirs transversaux) sont uniformes (de même taille).

L'entrepôt dispose de Z préparateurs qui doivent préparer la prochaine vague de commandes. Soit $\mathcal{I} = \{1, \cdots, s, \cdots, S\}$ la liste qui énumère les items des commandes associés à cette vague. Chaque item $s \in \mathcal{I}$ est demandé en r_s unités. De plus, il est disponible à partir d'un sous-ensemble d'emplacements de collecte P_s en quantités limitées. Soit q_p^s la quantité de l'item s disponible à partir de l'emplacement $p \in P_s$.

Soit $\{o_1, o_2, \cdots, o_Z\}$ l'ensemble des dépôts de l'entrepôt, avec o_z étant le dépôt associé à la zone z. Soit $\mathcal{P}_z$ l'ensemble représentant le dépôt o_z ainsi que les différents emplacements de collecte dans la zone z à partir desquels des items de $\mathcal{I}$ peuvent être récupérés. Finalement,

on définit $d_{p,p'}$ comme la plus courte distance entre les emplacements p et $p' \in \mathcal{P}_z$.

Soit $\pi = \{\pi_z, z \in \{1, \cdots, Z\}\}$ une représentation d'une solution de notre problème, avec $\pi_z = (o_z, \pi_z^1, \cdots, \pi_z^i, \cdots, \pi_z^{N_z}, o_z)$ étant la tournée associée au préparateur z. Dans π_z, o_z correspond au dépôt de la zone z et $\{\pi_z^i \in \mathcal{P}_z, i \in \{1, \cdots, N_z\}\}$ correspond aux emplacements de collecte sélectionnés pour la zone z triés selon l'ordre de visite. π est dite réalisable si elle satisfait les conditions suivantes :

— la demande de chaque item dans la liste $\mathcal{I}$ est satisfaite à travers les emplacements visités par les préparateurs dans leur tournée

— la quantité de l'item s récupérée à partir de chaque emplacement p ne dépasse pas q_p^s. Il est à noter que, puisque la quantité d'un item s disponible à partir d'un emplacement p est limitée, plusieurs emplacements pourraient être visités pour satisfaire la demande de s.

Finalement, l'objective du problème est de minimiser le *makespan* $Cmax$, c.-à-d. la distance de la plus longue tournée. Ce dernier est défini par l'équation 6.1.

$$Cmax = max_z\{\pi_z^{opt}\} \tag{6.1}$$
$$avec \;\; \pi_z^{opt} = argmin_{\pi_z}\left(f(\pi_z) = d_{o_z,\pi_z^1} + \sum_{i=1}^{N_z-1} d_{\pi_z^i,\pi_z^{i+1}} + d_{\pi_z^{N_z},o_z}\right).$$

6.3 Décomposition de Benders de type “logic-based”

Pour résoudre le problème dans le cas générique des entrepôts multi-blocs, nous proposons une procédure de décomposition de Benders de type “logic-based” (LBBD). LBBD est une généralisation de la procédure de décomposition de Benders classique pour des problèmes pour lesquels la dualité de la programmation linéaire ne peut être utilisée pour générer des coupes de Benders. Au lieu de cela, LBBD résout un problème dit *dual d'inférence*. La solution du problème dual d'inférence est utilisée pour générer des coupes qui sont ajoutées au problème maître. Notons que, contrairement à la méthode de décomposition de Benders classique, aucune procédure générique n'est disponible pour générer de telles coupes dans les décompositions de type “logic-based”. Les coupes doivent être définies selon le problème étudié [84].

Notre procédure correspond à une adaptation de celle introduite dans [92] pour le PRP-MS. Le problème est décomposé en un problème maître et un sous-problème. Dans le problème maître, on sélectionne les emplacements de collecte à visiter dans chaque zone z de telle sorte que la demande de chaque item dans $\mathcal{I}$ est satisfaite par les étagères accessibles à

partir des emplacements sélectionnés. Dans le sous-problème, un PRP est résolu pour chaque préparateur z avec les emplacements donnés par le problème maître. Dans la suite de cette section, nous décrivons la formulation du problème maître, le sous-problème, les coupes de Benders générées et l'implémentation de la procédure.

6.3.1 Problème maître

Dans la formulation du problème maître, un ensemble de *configurations non-dominées* $\mathcal{C}_{k,l}$ est défini pour chaque sous-allée (k, l). Cette ensemble correspond à tous les sous-ensembles d'emplacements de collecte qui peuvent être visités en traversant entièrement (k, l), en entrant et en sortant de (k, l) par le haut (*entrée-sortie par* $t_{(k,l)}$), par le bas (*entrée-sortie par* $b_{(k,l)}$) ou par les deux côtés. Dans notre formulation, on pose l'hypothèse que si un préparateur de commandes entre dans une sous-allée à partir de $t_{(k,l)}$ (resp. par $b_{(k,l)}$) et se rend à l'emplacement p, il est toujours profitable de s'arrêter dans tous les emplacements de collecte entre $t_{(k,l)}$ (resp. par $b_{(k,l)}$) et p pour récupérer des items de $\mathcal{I}$ disponibles à partir de ces emplacements. Ceci est dû au fait que les coûts de la fonction objectif dépendent uniquement de la distance parcourue en se déplaçant entre les emplacements de collecte. Par conséquent, nous conservons uniquement les configurations qui suivent cette caractéristique et considérons les autres comme *dominées*. La figure 6.3 donne un exemple d'une sous-allée qui contient quatre étagères de chaque côté et l'ensemble des configurations possibles pour cette sous-allée. Pour chaque configuration, les emplacements de collecte à visiter sont représentés par des nœuds noirs et les extrémités de la sous-allée sont représentées par les nœuds oranges. Les deux dernières configurations sont des exemples de configurations dominées. Finalement, soit q_c^s la quantité de l'item s qui peut être récupérée en utilisant la configuration c. L'objectif du problème est alors de sélectionner au plus une configuration par sous-allée de sorte que la quantité de chaque item s qui peut être récupérée par les configurations sélectionnées satisfait la demande r_s .

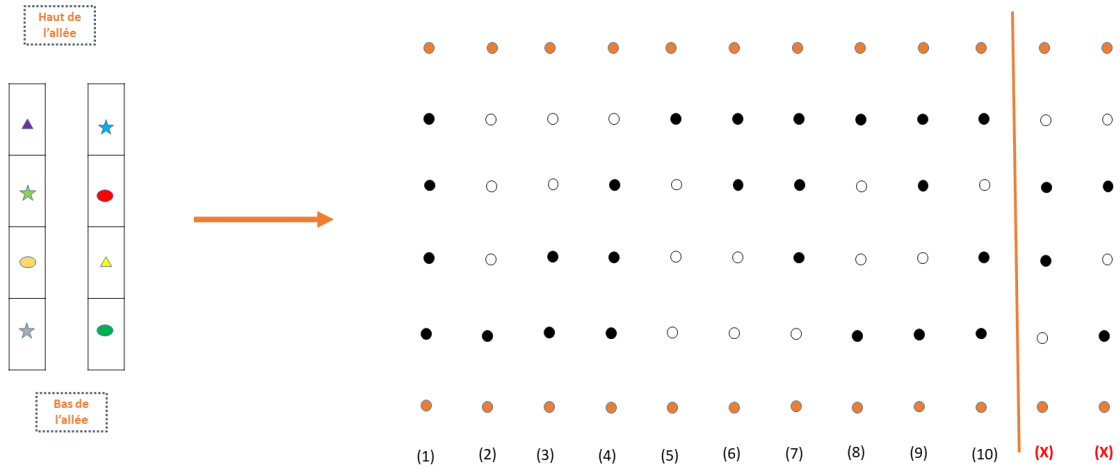


FIGURE 6.3 L'ensemble des configurations non-dominées et un exemple de deux configurations dominées pour une sous-allée avec quatre emplacements de collecte

La formulation du problème maître est renforcée par des techniques d'accélération algorithmiques, à savoir une fonction de génération de bornes inférieures et des inégalités valides (IVs). La fonction de génération de bornes inférieures permet d'approximer la valeur de $Cmax$ dans le problème maître en fonction des configurations sélectionnées. Elle définit, pour chaque zone z, une distance horizontale et une distance verticale. La distance horizontale pour la zone z correspond à la distance d'un aller-retour entre o_z et l'allée la plus éloignée à visiter dans la zone z. Pour la distance verticale, une distance minimale $\tilde{d}^c_{k,l}$ est définie pour chaque sous-allée (k,l) et chaque configuration c. Elle correspond à une borne inférieure sur la distance à parcourir dans le sous-problème pour la sous-allée (k,l) si la configuration c est sélectionnée. Notons qu'un calcul initial et un calcul amélioré de $\tilde{d}^c_{k,l}$ sont proposés dans [92], et les deux seront utilisés dans notre implémentation.

Les inégalités valides (IVs) sont des coupes qui permettent d'éliminer certaines solutions dominées dans le problème maître en imposant des restrictions dans la sélection des configurations. Elles comprennent des IVs *intra-bloc* et des IVs *inter-blocs*. Dans les IVs intra-bloc, nous exploitons le fait que pour un bloc k dans une zone z, si le préparateur entre dans une sous-allée (k,l) par le haut, alors il y a au moins deux autres sous-allées dans le bloc k de la zone z qui doivent être complètement traversées. Les IVs inter-blocs opèrent sur deux blocs successifs. Elles modélisent le fait que si un bloc $k+1$ est atteint dans une zone z, alors le bloc k de la zone z doit être complètement traversé au moins deux fois. Pour plus de détails sur la fonction de génération de bornes inférieures et les inégalités valides, on réfère le lecteur

intéressé à l'article [92]. La principale adaptation de la formulation donnée dans [92] à ce problème est que la fonction de génération de bornes inférieures et les inégalités valides sont définis pour chaque zone z.

Dans la suite de cette section, les ensembles, les paramètres et les variables sont donnés dans le tableau 6.1. Ensuite, la formulation du problème maître est décrite par les équations (6.2) - (6.15).

Tableau 6.1 Notation utilisée dans la formulation du problème maître

Ensembles	
$\mathcal{Z}$	l'ensemble des zones
$\mathcal{L}_z$	les allées associées à la zone z, avec L_z la dernière allée de la zone
$\mathcal{L}^d$	l'ensemble des allées contenant un dépôt. Le dépôt de chaque zone est situé en bas de la première allée de cette zone
$\mathcal{C}_{k,l}$	l'ensemble des configurations de la sous-allée (k,l)
$\check{\mathcal{C}}_{k,l}$	l'ensemble des configurations de la sous-allée (k,l) caractérisées par une entrée-sortie par le haut
$\mathcal{C}^{all}_{k,l}$	singleton incluant la configuration qui contient tous les emplacements à partir desquelles des items de $\mathcal{I}$ peuvent être collectés dans (k,l)
$\mathcal{C}^{all}_{k,z}$	l'ensemble des singletons du bloc k de la zone z ($\mathcal{C}^{all}_{k,z} = \bigcup_{l\in\mathcal{L}_z} \mathcal{C}^{all}_{k}$)
$\mathcal{P}^c_{k,l}$	le sous-ensemble d'emplacements de collecte qui doivent être visités dans (k,l) à travers la configuration c
Paramètres	
q^s_c	la quantité de l'item s qui peut être récupéré à travers la configuration c
$\bar{d}_l$	la plus courte distance entre l'allée l et o_z
$\tilde{d}^c_{k,l}$	la distance minimale pour parcourir l'allée (k,l) en utilisant c
Variables	
$x^c_{k,l} \in \{0,1\}$	vaut 1 si la configuration c est sélectionnée pour la sous-allée (k,l), 0 sinon
$\theta \in \mathbb{R}^+$	borne inférieure sur $Cmax$
$\lambda_z \in \mathbb{R}^+$	distance horizontale pour le préparateur z
$\psi_{k,l} \in \mathbb{R}^+$	distance verticale pour l'allée (k,l)
$\beta^d_{k,z} \in \{0,1\}$	vaut 1 si le bloc $k+1$ dans la zone z est atteint dans une sous-allée $(k+1,l)$ avec $l \in \mathcal{L}^d$, 0 sinon
$\beta_{k,z} \in \{0,1\}$	vaut 1 si le bloc $k+1$ dans la zone z est atteint dans une sous-allée $(k+1,l)$ avec $l \notin \mathcal{L}^d$, 0 sinon

$$\min \theta \tag{6.2}$$

$$\sum_{c \in \mathcal{C}_{k,l}} x^c_{k,l} \leq 1 \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \tag{6.3}$$

$$\sum_{k=0}^{K-1} \sum_{l=0}^{L-1} \sum_{c \in \mathcal{C}_{k,l}} q^s_c \cdot x^c_{k,l} \geq r_s \qquad \forall s \in \mathcal{I} \tag{6.4}$$

$$2 \cdot \sum_{c \in \mathcal{C}_{k,l}} \bar{d}_l \cdot x^c_{k,l} \leq \lambda_z \qquad \forall k \in \mathcal{K}, \forall z \in \mathcal{Z}, \forall l \in \mathcal{L}_z \tag{6.5}$$

$$\sum_{c \in \mathcal{C}_{k,l}} \tilde{d}^c_{k,l} \cdot x^c_{k,l} \leq \psi_{k,l} \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L} \tag{6.6}$$

$$\sum_{k=0}^{K-1} \sum_{l \in \mathcal{L}_z} \psi_{k,l} + \lambda_z \leq \theta \qquad \forall z \in \mathcal{Z} \tag{6.7}$$

$$\sum_{l' \in \mathcal{L}_z, l' \neq l} \sum_{c \in \mathcal{C}^{all}_{k,l'}} x^c_{k,l'} \geq 2 \cdot \sum_{c \in \check{\mathcal{C}}_{k,l}} x^c_{k,l} \qquad \forall k \in \mathcal{K}, \forall z \in \mathcal{Z}, \forall l \in \mathcal{L}_z \tag{6.8}$$

$$\sum_{c \in \mathcal{C}_{k+1,l}} x^c_{k+1,l} \leq \beta^d_{k,z} \qquad \forall k \in \{0, \cdots, K-2\} \tag{6.9}$$

$$\forall z \in \mathcal{Z}, \forall l \in \mathcal{L}_z \cap \mathcal{L}^d$$

$$\sum_{l \in \mathcal{L}_z} \sum_{c \in \mathcal{C}^{all}_{k,l}} x^c_{k,l} \geq \beta^d_{k,z} \qquad \forall k \in \{0, \cdots, K-2\}, \forall z \in \mathcal{Z} \tag{6.10}$$

$$\sum_{c \in \mathcal{C}_{k+1,l}} x^c_{k+1,l} \leq \beta_{k,z} \qquad \forall k \in \{0, \cdots, K-2\} \tag{6.11}$$

$$\forall z \in \mathcal{Z}, \forall l \in \mathcal{L}_z \setminus \mathcal{L}^d$$

$$\sum_{l \in \mathcal{L}_z} \sum_{c \in \mathcal{C}^{all}_{k,l}} x^c_{k,l} \geq 2 \cdot \beta_{k,z} \qquad \forall k \in \{0, \cdots, K-2\}, \forall z \in \mathcal{Z} \tag{6.12}$$

$$\beta_{k,z} \geq \beta_{k+1,z} \qquad \forall k \in \{0, \cdots, K-3\}, \forall z \in \mathcal{Z} \tag{6.13}$$

$$\textit{coupes générées précédemment} \qquad \forall z \in \mathcal{Z} \tag{6.14}$$

$$x^c_{k,l}, \beta_{k,z}, \beta^d_{k,z} \psi_{k,l} \in \{0,1\} \qquad \forall l \in \mathcal{L}, \forall c \in \mathcal{C}_{k,l} \tag{6.15}$$

$$\theta, \lambda_z \in \mathbb{R}^+ \qquad \forall k \in \mathcal{K}, \forall z \in \mathcal{Z}, \forall l \in \mathcal{L}_z$$

La fonction objective est définie par l'équation (6.2). Les contraintes (6.3) permettent de sélectionner au plus une configuration pour chaque sous-allée (k, l). Les contraintes (6.4) garantissent que la demande de chaque item dans $\mathcal{I}$ peut être satisfaite à travers les configurations sélectionnées. Les contraintes (6.5) - (6.7) modélisent la fonction de génération de bornes inférieures. Les contraintes (6.8) correspondent aux coupes intra-bloc. Elles garantissent que si une configuration c correspondant à une entrée-sortie de (k, l) par $t_{(k,l)}$ est

sélectionnée dans une zone z, alors au moins deux autres sous-allées dans le bloc k de la zone z doivent être couvertes par des configurations $\in \mathcal{C}^{all}_{k,z}$. Les contraintes (6.9) - (6.13) modélisent les coupes inter-blocs. Pour une zone z et un bloc k donnés, on distingue deux cas. Dans le premier cas, le bloc supérieur $(k+1)$ de cette zone est atteint dans l'allée contenant le dépôt (contraintes (6.9)). Par conséquent, au moins une sous-allée dans le bloc k de la zone z doit être couverte par une configuration appartenant à $\mathcal{C}^{all}_{k,z}$ (contraintes (6.10)). Dans le deuxième cas, le bloc $k+1$ de la zone z est atteint dans une autre allée que celle contenant le dépôt (contraintes (6.11) et (6.12)). Alors, au moins deux sous-allées dans le bloc k de la zone z doivent être couvertes par des configurations appartenant à $\mathcal{C}^{all}_{k,z}$ (contraintes (6.13)). Les contraintes (6.14) correspondent aux coupes d'optimalité générées lors de la résolution du sous-problème dans les itérations précédentes et seront discutées dans la prochaine sous-section. Enfin, les contraintes (6.15) définissent le domaine des variables de décision.

6.3.2 Sous-problème, coupes de Benders et implémentation

Dans cette section, nous présentons comment les coupes de Benders sont générées à chaque itération. D'abord, notons que pour tout sous-ensemble de configurations retourné par le problème maître, le sous-problème est toujours réalisable. Par conséquent, on génère uniquement des coupes d'optimalité. Soit $\bar{\mathcal{C}}_z$ l'ensemble des configurations retournées par le problème maître pour la zone z ($\bar{\mathcal{C}}_z = \{(c,k,l) : c \in \mathcal{C}_{k,l}\ \&\ x^c_{k,l} = 1, l \in \mathcal{L}_z\}$). Soit $\bar{\mathcal{P}}_z$ l'ensemble des emplacements associés aux configurations dans $\bar{\mathcal{C}}_z$ ($\bar{\mathcal{P}}_z = \{p \in \mathcal{P}^c_{k,l} : (c,k,l) \in \bar{\mathcal{C}}_z\}$). Pour chaque zone z, nous résolvons un PRP avec les nœuds suivants $\{o_z\} \cup \bar{\mathcal{P}}_z$. Soit $\bar{\theta}_z$ la distance de la tournée optimale pour la zone z. La coupe d'optimalité associée à la zone z est alors (adaptée de [92]) :

$$\theta \geq \bar{\theta}_z - \sum_{(c,k,l)\in\bar{\mathcal{C}}_z} \sum_{p\in\mathcal{P}^c_{k,l}} \left[(1 - x^c_{k,l}) \cdot 2 \cdot d_{o_z,p}\right] \qquad \forall z \in \mathcal{Z} \qquad (6.16)$$

Dans nos tests préliminaires, nous avons d'abord essayé une implémentation classique de LBBD dans laquelle nous résolvons de manière optimale le problème maître et le sous-problème, et nous générons Z coupes qui sont ajoutées au problème maître à chaque itération. Puisque le problème maître est résolu à l'optimum, nous avons calculé les $\tilde{d}^c_{k,l}$ avec la version améliorée. Nous avons observé que, dans certains cas, le problème maître consomme une partie importante du temps de calcul à chaque itération sans générer de coupes décisives. Nous avons ensuite essayé une implémentation de type *branch-and-check* dans laquelle les coupes sont générées à chaque solution entière du problème maître. Dans cette version, nous

avons calculé les $\tilde{d}^c_{k,l}$ en utilisant la version initiale pour garantir que la valeur de θ à chaque solution entière du problème maître est plus petite ou égale à la valeur de la solution renvoyée par le sous-problème. Nous avons constaté que le nombre de coupes générées pour certaines instances était trop grand. Nous avons aussi essayé une autre version dans laquelle une limite de temps est fixée pour le problème maître à chaque itération, qui est généré aléatoirement dans l'intervalle $[t_{min}, t_{max}]$. Cette approche permet de fournir des solutions de meilleure qualité que les deux premières versions pour certaines instances. Finalement, nous avons décidé d'implémenter une version parallélisée de la méthode dans laquelle nous exécutons la première version (implémentation classique) et la dernière version (implémentation avec limite de temps pour le problème maître) et arrêtons l'algorithme lorsqu'une solution optimale est trouvée. Puisque les instances sur lesquelles nous avons exécuté nos méthodes sont définies sur des entrepôts 1-bloc (voir la Section 6.5), nous avons utilisé le PLNE de [35] pour résoudre les PRPs à chaque itération. Cependant, de nombreuses méthodes efficaces existent dans la littérature pour résoudre le PRP pour les entrepôts multi-blocs, et peuvent être intégrées dans notre procédure pour résoudre les instances multi-blocs (voir la Section 4.1).

6.4 Formulation compacte pour les entrepôts disposés en un bloc

Dans cette section, nous proposons une formulation compacte spécialisée pour les entrepôts qui sont disposés en un bloc. La formulation est adaptée d'une méthode de l'état de l'art [35] pour résoudre le PRP-MS pour ce type d'entrepôts.

Dans [35], les auteurs exploitent certaines propriétés relatives aux entrepôts à un bloc, introduites dans [27]. Les deux premières sont les suivantes :

- Dans une tournée optimale, deux allées successives l et $l+1$ sont connectées selon des quatre manières (transitions) suivantes : deux arêtes entre $t_{(0,l)}$ et $t_{(0,l+1)}$ (deux connexions par le haut), deux arêtes entre $b_{(0,l)}$ et $b_{(0,l+1)}$ (deux connexions par le bas), une arête entre $t_{(0,l)}$ et $t_{(0,l+1)}$ et une autre entre $b_{(0,l)}$ et $b_{(0,l+1)}$ (une connexion des deux côtés) et deux arêtes entre $t_{(0,l)}$ et $t_{(0,l+1)}$ et deux autres entre $b_{(0,l)}$ et $b_{(0,l+1)}$ (deux connexions des deux côtés).
- La génération d'une tournée connectée avec un degré pair en haut et en bas de chaque allée est une condition suffisante pour éviter la génération de sous-tours isolés.

L'idée générale de cette formulation est de définir un ensemble de variables et de contraintes qui permettent de : 1. sélectionner les emplacements de collecte à visiter de telle sorte à satisfaire la demande de chaque item ; 2. construire la tournée en parcourant les allées de

collecte de gauche à droite, en décidant comment traverser chaque allée qui contient des emplacements de collecte sélectionnés et en connectant deux allées successives avec l'une des quatre transitions décrites ci-dessus. La première allée de la tournée correspond à l'allée contenant le dépôt et la dernière allée à celle contenant un emplacement de collecte à visiter la plus à droite. Les variables sont gérées de telle manière que la tournée partielle entre la première allée et une allée donnée l satisfait les conditions suivantes :

1. Le degré en $b_{(0,l)}$ et en $t_{(0,l)}$ pour chaque chaque allée l incluse dans cette tournée est pair (éventuellement de degré 0).
2. La tournée partielle est soit connectée ou constituée de deux composantes connexes. C'est une autre propriété, introduite dans [27], qui constitue une condition nécessaire et suffisante pour que la tournée finale ne dispose pas de sous-tours isolés.

Dans notre formulation, nous généralisons cette idée en générant une tournée pour chaque zone z et en adaptant la formulation pour prendre en compte la fonction objectif (minimisation du *makespan*) au lieu de la minimisation de la distance totale parcourue.

Formellement, nous définissons pour chaque allée $l \in \mathcal{L}$, $\mathcal{P}_l = \{1, \cdots, P_l\}$, l'ensemble des emplacements de collecte qui contiennent des items de $\mathcal{I}$ ordonnés de haut en bas et z_l la zone associée à l'allée l. La formulation utilise six groupes de variables pour chaque allée l :

— Le premier groupe décide si l'allée l est *atteinte* par le préparateur z_l . On dit qu'une allée l est *atteinte* si le préparateur entre dans cette allée pour récupérer des items ou s'il passe par $b_{(0,l)}$ ou $t_{(0,l)}$ pour accéder à une autre allée $l' > l$.
— Le deuxième groupe gère la manière de connecter les allées l et $l+1$, c.-à-d. avec l'une des quatre transitions présentées ci-dessus. Ce groupe est défini pour toutes les allées d'une zone sauf la dernière.
— Le troisième groupe gère la manière de traverser l'allée l, c.-à-d. une traversée complète, une entrée-sortie par le haut, une entrée-sortie par le bas ou une entrée-sortie par les deux côtés de l'allée. Il est à noter que dans la formulation de [35], une variable correspondant à une traversée complète dans les deux sens de l'allée l est considérée. Cependant, elle est uniquement nécessaire dans le cas d'un entrepôt contenant des allées qui ne sont pas uniformes. Ce qui n'est pas le cas dans le problème que nous considérons.
— Le quatrième groupe contrôle le degré de connexion en haut (resp. en bas) de l'allée l. Il s'agit de variables entières qui exploitent le fait que si le degré de $t_{(0,l)}$ (resp. de $b_{(0,l)}$) est pair, alors le degré divisé par deux est égal à un nombre entier.
— Le cinquième groupe gère le nombre de composantes de la tournée partielle de o_z à l.

— Le dernier groupe sélectionne des emplacements de collecte à visiter dans l'allée l.

Les paramètres et les variables utilisés dans le modèle sont répertoriés dans le tableau 6.2. La formulation PLNE est donnée par les équations (6.17) - (6.40).

Tableau 6.2 Notation utilisée dans la formulation compacte

Paramètres	
$q^s_{l,p}$	la quantité de l'item s accessible à partir de p dans l'allée l
c	la distance entre deux allées successives
$c^{\vert}$	la longueur d'une allée
$\check{c}_{l,p}$	la distance associée à une entrée-sortie par $t_{(0,l)}$ jusqu'à p
$\hat{c}_{l,p}$	la distance associée à une entrée-sortie par $b_{(0,l)}$ jusqu'à p
Variables	
$\widetilde{y}_l$	vaut 1 si l'allée l est atteinte, 0 sinon.
$\bar{\bar{y}}_l$	vaut 1 si les allées l et $l+1$ sont connectées deux fois par le haut, 0 sinon
$\underline{\underline{y}}_l$	vaut 1 si les allées l et $l+1$ sont connectées deux fois par le bas, 0 sinon
$\underline{\bar{y}}_l$	vaut 1 si les allées l et $l+1$ sont connectées une fois par le haut et une fois par le bas, 0 sinon
$\underline{\underline{\bar{\bar{y}}}}_l$	vaut 1 si les allées l et $l+1$ sont connectées deux fois par le haut et deux fois par le bas, 0 sinon
$\hat{y}_{l,p}$	vaut 1 si une entrée-sortie par $b_{(0,l)}$ jusqu'à p est faite, 0 sinon
$\check{y}_{l,p}$	vaut 1 si une entrée-sortie par $t_{(0,l)}$ jusqu'à p est faite, 0 sinon
$y^{\vert}_l$	vaut 1 si l'allée l est complètement traversée, 0 sinon
$\mathring{\pi}_l$	variable entière qui est égale au degré de $t_{(0,l)}$ divisé par 2
π_l	variable entière qui est égale au degré de $b_{(0,l)}$ divisé par 2
$\mathring{\tau}_l$	vaut 0 si la tournée partielle de o_{z_l} à l est connectée, 1 si elle inclue deux composantes connexes. Il est à noter que les contraintes sont gérées tel que $\tau_l = 0$ pour toute allée l située entre la dernière allée atteinte et la dernière allée de la zone
$x_{l,p}$	vaut 1 si l'emplacement p de l'allée l est visité, 0 sinon

$$\min Cmax \tag{6.17}$$

$$\widetilde{y}_l \geq x_{l,p} \qquad \forall l \in \mathcal{L}, \forall p \in \mathcal{P}_l \tag{6.18}$$

$$\widetilde{y}_l = 1 \qquad \forall l \in \mathcal{L}^d \tag{6.19}$$

$$\widetilde{y}_l \geq \widetilde{y}_{l+1} \qquad \forall z \in \mathcal{Z}, \forall l \in \mathcal{L} \setminus \{L_z\} \tag{6.20}$$

$$\bar{\bar{y}}_l + \underline{y}_l + \bar{\underline{y}}_l + \bar{\bar{\underline{y}}}_l = \widetilde{y}_{l+1} \qquad \forall z \in \mathcal{Z}, \forall l \in \mathcal{L} \setminus \{L_z\} \tag{6.21}$$

$$\sum_{l \in \mathcal{L}} \sum_{p \in \mathcal{P}_l} q^s_{l,p} \cdot x_{l,p} \geq r_s \qquad \forall s \in \mathcal{I} \tag{6.22}$$

$$y^{\downarrow}_l + \sum_{p' \in \mathcal{P}_l; p' \geq p} \check{y}_{l,p} + \sum_{p' \in \mathcal{P}_l; p' \leq p} \hat{y}_{l,p} \geq x_{l,p} \qquad \forall l \in \mathcal{L}, \forall p \in \mathcal{P}_l \tag{6.23}$$

$$\underline{y}_{l-1} + \bar{\underline{y}}_{l-1} + \bar{\bar{\underline{y}}}_{l-1} + \underline{y}_l + \bar{\underline{y}}_l + \bar{\bar{\underline{y}}}_l \geq \hat{y}_{l,p} \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d, \forall p \in \mathcal{P}_l \tag{6.24}$$

$$\bar{\bar{y}}_{l-1} + \bar{\underline{y}}_{l-1} + \bar{\bar{\underline{y}}}_{l-1} + \bar{\bar{y}}_l + \bar{\underline{y}}_l + \bar{\bar{\underline{y}}}_l \geq \check{y}_{l,p} \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d, \forall p \in \mathcal{P}_l \tag{6.25}$$

$$\bar{\bar{y}}_l + \bar{\underline{y}}_l + \bar{\bar{\underline{y}}}_l \geq \check{y}_{l,p} \qquad \forall l \in \mathcal{L}^d, \forall p \in \mathcal{P}_l \tag{6.26}$$

$$2\bar{\bar{y}}_{l-1} + \bar{\underline{y}}_{l-1} + 2\bar{\bar{\underline{y}}}_{l-1} + 2\bar{\bar{y}}_l + \bar{\underline{y}}_l + 2\bar{\bar{\underline{y}}}_l + y^{\downarrow}_l = 2\overset{\circ}{\pi}_l \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d \tag{6.27}$$

$$2\bar{\bar{y}}_l + \bar{\underline{y}}_l + 2\bar{\bar{\underline{y}}}_l + y^{\downarrow}_l = 2\overset{\circ}{\pi}_l \qquad \forall l \in \mathcal{L}^d \tag{6.28}$$

$$2\underline{y}_{l-1} + \bar{\underline{y}}_{l-1} + 2\bar{\bar{\underline{y}}}_{l-1} + 2\underline{y}_l + \bar{\underline{y}}_l + 2\bar{\bar{\underline{y}}}_l + y^{\downarrow}_l = 2\underset{\circ}{\pi}_l \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d \tag{6.29}$$

$$2\underline{y}_l + \bar{\underline{y}}_l + 2\bar{\bar{\underline{y}}}_l + y^{\downarrow}_l = 2\underset{\circ}{\pi}_l \qquad \forall l \in \mathcal{L}^d \tag{6.30}$$

$$\bar{\bar{\underline{y}}}_l \leq \tau_l \qquad \forall l \in \mathcal{L}^d \tag{6.31}$$

$$\underline{y}_{l-1} + \bar{\bar{y}}_{l-1} + \bar{\bar{\underline{y}}}_l \leq \tau_l + 1 \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d \tag{6.32}$$

$$\tau_{l-1} - y^{\downarrow}_l \leq \tau_l \qquad \forall l \in \mathcal{L} \setminus \mathcal{L}^d \tag{6.33}$$

$$\tau_l \leq \bar{\bar{\underline{y}}}_l \qquad \forall l \in \mathcal{L} \tag{6.34}$$

$$y^{\downarrow}_l + \underline{y}_l + \bar{\bar{\underline{y}}}_l \geq \bar{\bar{y}}_l \qquad \forall l \in \mathcal{L}^d \tag{6.35}$$

$$Cmax \geq \sum_{l \in \mathcal{L}_z} (2c\bar{\bar{y}}_l + 2c\underline{y}_l + 2c\bar{\underline{y}}_l + 4c\bar{\bar{\underline{y}}}_l) + c^{\downarrow} y^{\downarrow}_l) \qquad \forall z \in \mathcal{Z} \tag{6.36}$$

$$+ \sum_{l \in \mathcal{L}_z} \sum_{p \in \mathcal{P}_l} (\check{c}_{l,p} \check{y}_{l,p} + \hat{c}_{l,p} \hat{y}_{l,p})$$

$$\bar{\bar{y}}_l, \underline{y}_l, \bar{\underline{y}}_l, \bar{\bar{\underline{y}}}_l, \tau_l = 0 \qquad \forall z \in \mathcal{Z}, l = L_z \tag{6.37}$$

$$\widetilde{y}_l, y^{\downarrow}_l, \bar{\bar{y}}_l, \underline{y}_l, \bar{\underline{y}}_l, \bar{\bar{\underline{y}}}_l, \tau_l \in \{0,1\} \qquad \forall l \in \mathcal{L} \tag{6.38}$$

$$x_{l,p}, \check{y}_{l,p}, \hat{y}_{l,p} \in \{0,1\} \qquad \forall l \in \mathcal{L}, \forall p \in \mathcal{P}_l \tag{6.39}$$

$$\overset{\circ}{\pi}_l, \underset{\circ}{\pi}_l \in \mathbb{N} \qquad \forall l \in \mathcal{L} \tag{6.40}$$

(6.17) définit la fonction objectif. Les contraintes (6.18) - (6.20) identifient les allées qui doivent être atteintes : les allées contenant des emplacements à visiter (6.18), les allées contenant un dépôt (6.19) et les allées entre le dépôt d'une zone et la dernière allée contenant un emplacement à visiter dans cette zone (6.20). Les contraintes (6.21) connectent une allée atteinte $l+1$ et l'allée précédente l avec l'une des quatre transitions possibles. Les contraintes (5.20) garantissent que la demande de chaque item de la liste est satisfaite. Les contraintes (6.23) garantissent que chaque allée contenant un emplacement p à visiter doit être traversée de telle manière que p soit visité. Les contraintes (6.24) - (6.26) garantissent que chaque allée, à l'exception de celles contenant un dépôt, qui est traversée par une entrée-sortie par le haut (resp. par le bas), doit être connectée à l'allée précédente ou à l'allée suivante (ou aux deux) en haut (resp. en bas) des allées. Les contraintes (6.27) - (6.30) assurent que $t_{(0,l)}$ (resp. $b_{(0,l)}$), pour chaque allée l, sont de degré pair ou 0. En effet, elles définissent les variables $\overset{\circ}{\pi}_l$ (resp. π_l) en les liant aux variables qui impliquent des connexions en haut (resp. en bas) de l'allée l et en garantissant qu'elles prennent des valeurs strictement positives si le nombre de connexions en $t_{(0,l)}$ (resp. en $b_{(0,l)}$) est strictement positif et pair. Les variables impliquant des connexions en $t_{(0,l)}$ (resp. $b_{(0,l)}$) sont $\bar{\bar{y}}_l$, $\underline{\bar{y}}_l$, $\underline{\bar{\bar{y}}}_l$ et $y_l^{\downarrow}$ (resp. $\underline{\underline{y}}_l$, $\underline{\bar{y}}_l$, $\underline{\bar{\bar{y}}}_l$ et $y_l^{\downarrow}$). Les variables correspondant à une entrée-sortie par le haut (resp. par le bas) de l'allée l sont ignorées, car elles impliquent deux connexions à chaque fois, ce qui n'affecte pas la parité du degré de $t_{(0,l)}$ (resp. de $b_{(0,l)}$). Le nombre de composantes de la tournée partielle est géré en modélisant les différentes situations qui peuvent se produire lors de l'ajout de connexions horizontales ou verticales. Notez que pour une allée l de la zone z, la tournée partielle inclut deux composantes connexes uniquement dans l'une des deux situations suivantes : 1. $l \in \mathcal{L}_d$ et $\underline{\bar{\bar{y}}}_l = 1$; 2. nous passons de la configuration $\bar{\bar{y}}_{l-1} = 1$ (ou $\underline{\underline{y}}_{l-1} = 1$) à la configuration $\underline{\bar{\bar{y}}}_l = 1$. La première situation est modélisée par les contraintes (6.31) alors que la seconde situation est modélisée par des contraintes (6.32). Les contraintes (6.33) permettent de propager le nombre de composantes en forçant le passage de deux composantes connexes en $l-1$ (c.-à-d. $\tau_{l-1} = 1$) à une tournée connectée en l (c.-à-d. $\tau_l = 0$) si l'allée l est complètement traversée. Les contraintes (6.34) garantissent que pour une zone donnée z, deux allées consécutives l et $l+1$ sont connectées avec deux connexions en haut et deux connexions en bas (c.-à-d. $\underline{\bar{\bar{y}}}_l = 1$) tant que la tournée partielle de o_z jusqu'à l comprend deux composantes connexes. Les contraintes énumérées jusqu'à présent permettent d'atteindre chaque allée contenant un dépôt, mais ne forcent pas la visite explicite du dépôt. En effet, on peut se ramener à une situation dans laquelle on connecte une allée $l \in \mathcal{L}_d$ avec l'allée suivante uniquement par le haut (c.-à-d. $\bar{\bar{y}}_l = 1$) sans violer aucune des contraintes précédentes. Les contraintes (6.35) permettent d'éliminer cette situation. Les contraintes (6.36) définissent la variable $Cmax$ comme étant la distance de la plus longue tournée parmi les différentes zones. Finalement, les contraintes (6.37) - (6.40) défi-

nissent le domaine des variables de décision. Notons que dans la formulation décrite dans [35], le dépôt peut être situé en bas ou en haut de n'importe quelle allée de la zone de collecte et la longueur des allées n'est pas nécessairement uniforme. Nous adaptons donc la formulation originale à notre problème en supprimant les variables et contraintes non pertinentes.

6.5 Résultats expérimentaux

Dans cette section, nous testons les deux méthodes sur un ensemble d'instances générées comprenant différents paramètres. De plus, nous étudions l'impact de passer d'une politique de stockage traditionnelle à une politique de stockage à étagères mixtes dans les entrepôts découpés en zones. Enfin, nous étudions l'impact de l'augmentation du nombre de zones sur la performance du processus de préparation des commandes. Dans ce qui suit, la méthode de décomposition de Benders de type "logic-based" est dénotée `LBBD` tandis que le PLNE compact est dénoté `PLNE-1B`.

Tous les calculs ont été effectués sur un ordinateur portable 64 bits équipé d'un processeur Intel Core i5-6300U (2,40 GHz), de 16 gigaoctets de RAM et sous un système d'exploitation Windows 10. Toutes les méthodes ont été implémentées en C++ (Visual Studio 2019), en utilisant IBM ILOG CPLEX Optimizer 12.8 pour résoudre les PLNE. Enfin, t_{min} a été fixé à $2s$, t_{max} à $15s$ et une limite de temps $TL = 300s$ a été imposée pour les deux méthodes.

6.5.1 Analyse de performance

L'ensemble des instances est défini sur un entrepôt disposé en un bloc. Il comprend deux groupes : Dans le premier groupe, l'entrepôt est découpé en deux zones et dispose de 10 allées avec 30 emplacements de collecte par allée. Dans le second groupe, l'entrepôt est découpé en quatre zones et dispose de 30 allées avec 60 emplacements de collecte par allée. Dans les deux groupes, la cardinalité de la liste de collecte $|\mathcal{I}| \in \{30, 50\}$. De plus, un facteur γ est utilisé pour contrôler la duplication des items dans l'entrepôt $\gamma \in \{5, 10\}$.

Afin de répartir les items dans les différentes étagères, nous reproduisons le même mécanisme que celui utilisé dans [35]. D'abord, le nombre total d'items stockés dans l'entrepôt, noté ξ, est fixé comme suit : $\xi = max(|\mathcal{I}|, \lceil L \cdot R/\gamma \rceil)$. Ensuite, les items sont regroupés en catégories en utilisant une classification ABC. La classe A comprend 20% des items, la classe B 30% et la classe C les 50% restants. Une unité de chaque item est ensuite affectée à une étagère associée à un emplacement de collecte choisi aléatoirement pour assurer la disponibilité de chaque item dans l'entrepôt. Finalement, les étagères associées aux autres emplacements

de collecte reçoivent un des items de la classe A, B ou C avec une probabilité de 80%, 15% ou 5%, respectivement. La quantité de chaque item disponible à partir de chaque emplacement de collecte est générée uniformément de l'ensemble $\{1, 2, 3\}$.

La liste $\mathcal{I}$ est construite en sélectionnant aléatoirement des items dans les classes A, B et C avec une probabilité de 80%, 15% et 5%, respectivement. Enfin, la demande de chaque item $s \in \mathcal{I}$ est générée aléatoirement comme un nombre entier dans l'intervalle $[1, min(6, \bar{q}_s)]$, avec $\bar{q}_s$ étant la quantité totale de l'item s stockée dans l'entrepôt. Cinq réplications sont générées pour chaque combinaison de valeur des paramètres, ce qui donne 40 instances au total ($= 2 \times 2 \times 2 \times 5$).

TABLEAU 6.3 Les performances de `LBBD` et `PLNE-1B`

	`PLNE-1B`		`LBBD`		δ
$(\lvert\mathcal{I}\rvert,\gamma)$	#opt	CPU	#opt	CPU	
$Z = 2$					
(30,5)	(5/5)	0.44	(5/5)	1.25	1
(30,10)	(5/5)	0.2	(5/5)	3.94	0
(50,5)	(5/5)	0.35	(5/5)	2.23	1
(50,10)	(5/5)	0.33	(5/5)	2.41	0
$Z = 4$					
(30,5)	(5/5)	6.4	(5/5)	2.76	5
(30,10)	(5/5)	34.17	(5/5)	74.15	2
(50,5)	(5/5)	9.44	(5/5)	6.28	3
(50,10)	(5/5)	19.407	(5/5)	20.51	4

Les principaux résultats sont répertoriés dans le tableau 6.3. Ils incluent :

— #opt : le nombre d'instances résolues à l'optimum pour chaque méthode et chaque combinaison de valeur des paramètres

— CPU : le temps d'exécution moyen pour chaque méthode et chaque combinaison de valeur des paramètres

— δ : le nombre d'instances pour lesquelles `LBBD` est plus rapide que `PLNE-1B`

On observe que toutes les instances sont résolues à l'optimum par les deux méthodes. Pour les entrepôts découpés en deux zones, `PLNE-1B` résout toutes les instances en moins d'une seconde alors que `LBBD` consomme plus de temps CPU (2.46s en moyenne). De plus, `PLNE-1B` est plus rapide que `LBBD` dans 18/20 instances.

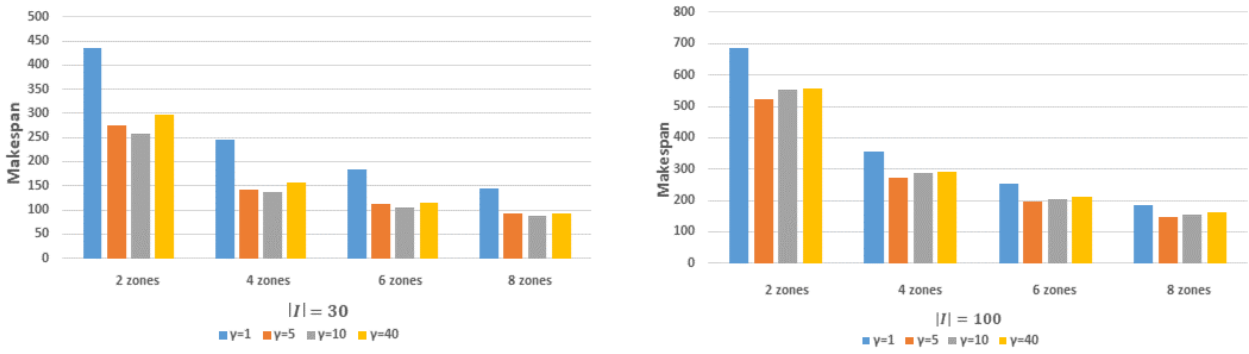
Pour les entrepôts découpés en quatre zones, PLNE-1B et LBBD sont compétitives, avec un léger avantage pour LBBD, qui est plus rapide que PLNE-1B dans 14/20 instances. Le temps CPU des deux méthodes est affecté par le degré de duplication des items dans l'entrepôt. Cependant, et pour les deux méthodes, aucune corrélation entre le temps CPU et la cardinalité de la liste $\mathcal{I}$ ne peut être établie.

Globalement, on conclut que les deux méthodes donnent de bons résultats sur l'ensemble des instances générées, avec un avantage pour PLNE-1B. Notons que PLNE-1B fonctionne exclusivement sur des entrepôts disposés en un bloc alors que LBBD peut être utilisée pour des entrepôts plus génériques (multi-blocs).

6.5.2 Perspectives managériales

Dans cette section, nous étudions les deux questions suivantes : 1. l'impact de la duplication des items dans l'entrepôt sur les performances de la préparation de commandes, 2. l'impact de l'augmentation du nombre de zones (préparateurs) sur les performances de la préparation de commandes. Pour cela, nous conduisons un ensemble d'expérimentations sur un entrepôt disposé en un bloc avec 30 allées et 40 emplacements par allée. Plus précisément, nous générons d'abord 20 instances pour chaque combinaison de valeur des paramètres suivants ($\gamma \in \{1, 5, 10, 40\}$, $|\mathcal{I}| \in \{30, 100\}$). Les instances avec $\gamma = 1$ simulent un entrepôt traditionnel dans lequel chaque item est disponible à partir d'un seul emplacement de collecte tandis que le nombre d'emplacements à partir desquels chaque item peut être récupéré croît avec l'augmentation de γ. Ensuite, nous résolvons notre problème pour chaque instance générée et chaque valeur de $Z \in \{2, 4, 6, 8\}$ en utilisant PLNE-1B. Les principaux résultats sont répertoriés dans les figures 6.4a et 6.4b. La figure 6.4a montre un diagramme dans lequel le *makespan* moyen pour chaque valeur de γ et Z est reporté pour les instances avec $|\mathcal{I}| = 30$. La figure 6.4b montre les mêmes informations pour $|\mathcal{I}| = 100$.

On observe à partir des deux figures que le passage d'un entrepôt traditionnel ($\gamma = 1$) aux entrepôts à étagères mixtes ($\gamma \in \{5, 10, 40\}$) conduit à une réduction considérable du *makespan*. L'impact semble être plus important pour les petites listes ($|\mathcal{I}| = 30$) que pour les grandes ($|\mathcal{I}| = 100$). En effet, le gain d'un stockage à étagère mixte par rapport à un stockage traditionnel réside dans le fait d'augmenter la probabilité de trouver les items d'une liste dans des allées proches les unes des autres (puisqu'ils sont disponibles dans plusieurs endroits), réduisant ainsi la longueur des tournées. Cependant, l'augmentation de la cardinalité des listes fait que les préparateurs doivent dans tous les cas visiter plusieurs allées pour récupérer les



(a) Instances avec $|\mathcal{I}| = 30$ (b) Instances avec $|\mathcal{I}| = 100$

FIGURE 6.4 *makespan* moyen pour les différents paramètres considérés

items de la liste, réduisant ainsi l'impact d'un stockage à étagères mixtes.

Une autre observation intéressante est que la duplication des items dans l'entrepôt en grandes proportions n'améliore pas nécessairement la performance du processus. En effet, nous observons que le passage de $\gamma = 5$ à $\gamma = 10$ (resp. de $\gamma = 10$ à $\gamma = 40$) a un effet minime (resp. un effet négatif) sur le *makespan* moyen pour les petites listes de collecte. Pour les grandes listes, l'observation est encore plus pertinente puisque l'augmentation de la valeur de γ fait croitre le *makespan* moyen. Cela est probablement dû au fait qu'en dupliquant les items plus qu'il en faut, la quantité de chaque item s stockée dans chaque étagère est tellement petite qu'elle oblige les préparateurs à visiter plus d'emplacements pour satisfaire r_s. On conclut donc que la gestion de la duplication des items dans l'entrepôt est un aspect important à considérer pour les gestionnaires lors de la mise en œuvre de la politique de stockage.

En ce qui concerne le nombre de zones, on observe globalement que le passage de deux zones à quatre zones réduit considérablement le *makespan* moyen, indépendamment de la cardinalité de $\mathcal{I}$ et du degré de duplication des items. La réduction est ensuite de moins en moins importante à chaque nouvelle zone ajoutée. Cette observation était prévisible puisque l'augmentation du nombre de zones au-delà d'un certain seuil conduit à la génération de tournées qui visitent un nombre limité d'emplacements (et donc d'allées) dans les zones nouvellement ajoutées, ce qui entraine des améliorations mineures. On conclut que la question de la détermination du nombre approprié de zones, c.-à-d. le nombre qui offre le meilleur compromis entre les performances du processus et les coûts opératoires, est pertinente d'un

point de vue managérial. Par exemple, il est suggéré de passer de deux à quatre zones, car le processus de préparation des commandes est quasiment deux fois plus rapide (449 en moyenne pour deux zones contre 236 pour quatre).

6.6 Conclusion

Dans ce chapitre, nous avons étudié un problème de préparation de commandes dans un entrepôt à étagères mixtes utilisant un zonage synchronisé. Étant donné un ensemble de zones et un préparateur opérant dans chaque zone, le problème consiste à concevoir la tournée de chaque préparateur pour collecter une liste d'items avec l'objectif de minimiser le *makespan*. De plus, vu que le stock de chaque item est divisé en petites unités qui sont dispersées un peu partout dans l'entrepôt, les emplacements de collecte pour satisfaire la demande de chaque item doivent être sélectionnés en plus de concevoir la tournée de collecte dans chaque zone.

Pour résoudre le problème, nous avons proposé deux méthodes, à savoir `LBBD` et `PLNE-1B`, adaptées de travaux antérieurs sur le PRP-MS. La première méthode (`LBBD`) est une décomposition de Benders de type "logic-based" dans laquelle on sélectionne les emplacements à partir d'où récupérer les items demandés dans le problème maître et on résout un PRP pour chaque zone dans le sous-problème. La deuxième méthode (`PLNE-1B`) est un PLNE compact spécialement conçu pour les entrepôts disposés en un bloc.

Dans la partie expérimentale, nous avons évalué la performance des solutions de chaque méthode. Les résultats montrent que les deux méthodes donnent des solutions optimales en des temps CPU raisonnables sur les instances considérées. De plus, nous avons étudié l'impact du passage d'un stockage traditionnel à un stockage à étagères mixtes sur le processus de préparation de commandes. Les résultats ont démontré que l'implémentation d'une politique de stockage à étagères mixtes réduit considérablement le *makespan*, en particulier pour les petites listes de collecte. C'est globalement le cas des entrepôts de commerce électronique dans lesquels des vagues de petites et urgentes commandes doivent être préparées à chaque fois. Nous avons également observé que l'augmentation du degré de duplication des items au-delà d'un certain seuil n'améliore pas nécessairement le processus de préparation des commandes et avons conclu que le contrôle du degré de duplication des items dans l'entrepôt est un aspect important à prendre en considération lors de la mise en œuvre d'une politique de stockage à étagères mixtes. Enfin, nous avons observé que le *makespan* moyen décroit de manière exponentielle en fonction du nombre de zones dans l'entrepôt.

Dans notre description du problème, nous avons supposé que la zone où opère chaque préparateur est fixe. Une perspective de recherche intéressante serait d'étendre nos modèles à une stratégie de zonage dynamique dans laquelle la taille de chaque zone peut être modifiée entre deux vagues successives de commandes. Dans ce cas-là, la taille de chaque zone devient un niveau de décision supplémentaire dans le problème. Aussi, il serait intéressant d'étudier la version *multi-trip* du problème dans laquelle chaque préparateur doit effectuer plusieurs tournées pour récupérer une vague de commandes.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Les services de livraison rapide émergent dans l'e-commerce, engendrant un nouveau pattern de demande constitué d'un grand nombre de petites commandes, chacune arrivant avec une date limite de livraison très courte. Afin de s'adapter à ce nouveau phénomène, une bonne gestion de la chaîne logistique, et notamment des activités liées à l'entreposage, passe par la mise en place de nouvelles techniques de stockage et de préparation des commandes. À travers cette thèse, nous avons proposé des modèles et des méthodes mathématiques pour optimiser le processus de préparation de commandes à la lumière de certaines de ces nouvelles pratiques et nous avons étudié leur impact. Dans ce chapitre, nous revenons sur l'ensemble des problèmes étudiés et des solutions proposées. De plus, nous identifions les limites associées à chaque méthode et tentons d'apporter de nouvelles perspectives de recherche.

7.1 Synthèse des travaux

Dans le chapitre 4, nous avons montré les performances qui peuvent être obtenues lors du processus de préparation des commandes dans les entrepôts d'e-commerce en implémentant une nouvelle stratégie de planification, à savoir la répartition d'une commande sur plusieurs préparateurs. Nous avons défini un problème dans lequel un ensemble de commandes, chacune composée d'un ensemble d'items et arrivant avec une date limite de préparation, doit être collecté par un ensemble de préparateurs de commandes munis de chariots de capacité limitée. L'objectif du problème est de construire des lots d'items, d'affecter les lots aux différents préparateurs de commandes, de séquencer les lots affectés à chaque préparateur et de construire la tournée associée à chaque lot de telle sorte que toutes les commandes soient préparées à temps et que la durée totale soit minimisée. Nous avons résolu le problème en utilisant une heuristique en deux étapes qui construit d'abord les lots d'items et les tournées associées à ces lots, et affecte ensuite les lots aux préparateurs de commandes et séquence les lots affectés à chaque préparateur. À travers une étude expérimentale sur 7290 instances de la littérature, nous avons démontré que la durée totale de collecte était réduite de 30% en moyenne (et pouvant aller jusqu'à 60%) en utilisant notre heuristique par rapport à la durée totale de collecte obtenue par une méthode de l'état de l'art élaborée pour la version du problème qui force la prise en charge de chaque demande par un seul préparateur dans une seule tournée. Finalement, nous avons conclu, à travers une analyse de la structure des solutions produites par chacune des méthodes, que la réduction du temps de collecte n'était pas due à une meilleure utilisation de la capacité des chariots, mais plutôt à la flexibilité

qu'offre la division des commandes sur les préparateurs dans la construction des lots d'items et de la tournée associée à chaque lot.

Dans le chapitre 5, nous avons étudié la planification d'une tournée d'un préparateur de commandes dans des entrepôts dits à étagères mixtes. Dans ce type d'entrepôts, le stock d'un item est divisé en petites quantités qui sont éparpillées dans différentes étagères dans l'entrepôt. Étant donné un ensemble d'items à récupérer en un ou plusieurs exemplaires, l'objectif du problème est de sélectionner les emplacements d'où récupérer les items et de construire la plus courte tournée qui commence et se termine au dépôt et visite les emplacements sélectionnés. Pour résoudre le problème, nous avons proposé une méthode de décomposition de type "logic-based" dans laquelle le problème maître sélectionne les emplacements de stockage à visiter et le sous-problème détermine la tournée de collecte. Contrairement aux méthodes disponibles dans la littérature, notre méthode est capable de s'adapter à des entrepôts disposés en plusieurs blocs. De plus, nous avons intégré un ensemble de nouvelles techniques d'amélioration incluant une fonction de génération de bornes inférieures et des inégalités valides. Nous avons vu que ces améliorations permettent de calculer de très bonnes bornes inférieures du problème et d'éliminer un grand nombre de solutions dominées rendant ainsi notre méthode beaucoup plus performante. Finalement, nous avons comparé notre méthode avec une adaptation d'une méthode de l'état de l'art, qui résout le problème de routage classique, aux entrepôts à étagères mixtes. Les résultats démontrent la supériorité de notre méthode.

Dans le chapitre 6, nous généralisons le problème défini dans le chapitre précédent en considérant que l'entrepôt est divisé en plusieurs zones disjointes et qu'un préparateur est affecté à chaque zone. Dans notre problème, une vague de commandes doit être récupérée dans la prochaine tournée des préparateurs. Ce cas est particulièrement pertinent dans un environnement dynamique, où les commandes arrivent en temps réel avec une date limite de préparation très courte. Chaque vague est donc constituée de commandes urgentes à collecter. L'objectif du problème est donc de sélectionner la zone et l'emplacement d'où récupérer les différents items et de construire le tour associé à chaque préparateur afin de minimiser la durée de collecte. Pour résoudre ce problème, nous avons adapté la méthode de décomposition proposée dans le chapitre précédent et proposé une formulation en MILP qui résout le problème pour le cas particulier des entrepôts à un seul bloc. Dans la partie expérimentale, nous avons démontré que l'implémentation d'une stratégie de stockage à étagères mixtes dans un entrepôt partitionné en zones pouvait considérablement augmenter les performances de la préparation de commandes par rapport à la stratégie de stockage classique.

7.2 Limitations de la solution proposée et améliorations futures

Dans le premier problème étudié au chapitre 4, nous avons supposé que le processus de collecte des items et le processus d'assemblage des items en commandes étaient réalisés d'une manière séquentielle et nous nous sommes principalement focalisés sur la collecte des items. Cependant, ces processus sont étroitement connectés et la considération d'une approche intégrée ouvre des perspectives de recherche intéressantes. De plus, plusieurs systèmes d'assemblage des items sont implémentés dans les entrepôts d'e-commerce. Il serait donc intéressant d'adapter ces modèles aux différents systèmes d'assemblage et de comparer la performance de ces systèmes.

Dans le chapitre 5, nous avons fait le choix d'implémenter, pour le cas d'entrepôts disposés en plusieurs blocs, le MILP de [10] pour résoudre le sous-problème de routage à chaque itération de notre méthode de décomposition. Nous avons constaté que pour certaines instances, la résolution de ce MILP nécessite un temps de calcul important pour retourner la solution optimale et, par conséquent, ne permet pas à la méthode de décomposition de faire plusieurs itérations. Une amélioration future serait d'intégrer le programme dynamique proposé par [10] pour résoudre le sous-problème de routage beaucoup plus rapidement. Concernant la définition du problème, nous avons considéré que l'ensemble des items à récupérer dans la prochaine tournée de chaque préparateur était déjà prédéfini et nous nous sommes principalement intéressés à la construction de la tournée de chaque préparateur indépendamment. Cependant, la construction des lots, l'affectation des lots aux préparateurs et la construction de la tournée pour récupérer chaque lot sont des niveaux de décisions étroitement liés et la généralisation de notre méthode à un problème plus général est une autre perspective de recherche intéressante. Ceci a été partiellement traité dans le chapitre 6 en considérant plusieurs préparateurs opérant chacun dans une zone prédéfinie et une vague de commandes à récupérer dans le prochain tour de chaque préparateur. Une critique qui peut être faite pour ce modèle est qu'il construit la prochaine tournée de chaque préparateur sans prendre en compte l'incertitude liée aux vagues suivantes de commandes. Dans un entrepôt à étagères mixtes, on peut être amené à trouver des solutions où le stock disponible aux alentours des dépôts est vidé dans les premières tournées et être contraint de construire de grandes tournées en fin de la journée. Bien que les réapprovisionnements de la zone de réserve vers la zone de collecte se font dans la journée dans ce type d'entrepôt, il serait tout de même intéressant d'étudier la version dynamique du problème dans laquelle les demandes arrivent en temps réel.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Dans cette thèse, nous nous sommes intéressés à des problèmes de planification de la préparation de commandes émergents dans les entrepôts de commerce électronique. Ces entrepôts ont la particularité de faire face à un nouveau pattern de demandes (un grand nombre de petites commandes, des délais de livraison serrés, un catalogue d'items riche) en mettant en place un ensemble de stratégies à l'échelle du stockage des items et de l'organisation de la préparation des commandes. Les travaux de cette thèse proposent des modèles et des méthodes qui prennent en compte ces nouvelles techniques et étudient leur pertinence.

Dans un premier temps, nous avons étudié un problème générique dans lequel les tournées d'un ensemble de préparateurs pour récupérer des commandes, arrivant avec des dates limites de préparation, doivent être planifiées. Dans notre définition du problème, nous avons introduit une nouvelle stratégie présente dans les entrepôts modernes, à savoir la décomposition des items d'une commande sur les préparateurs. Pour résoudre ce problème complexe, nous avons proposé une heuristique en deux étapes spécialement conçue pour tirer avantage du fait que les commandes peuvent être réparties sur plusieurs préparateurs. À travers un protocole expérimental rigoureux et des tests statistiques, nous avons prouvé que cette pratique permet de réduire considérablement le temps de collecte des commandes, validant ainsi sa pertinence.

Dans un deuxième temps, nous nous sommes intéressés à une autre stratégie implémentée par des entrepôts modernes, à savoir le stockage à étagères mixtes. À cet effet, nous avons étudié l'adaptation d'un problème classique dans la littérature de la préparation des commandes (PRP) au cas des entrepôts à étagères mixtes (PRP-MS). Nous avons introduit deux méthodes exactes (une décomposition de Benders de type "logic-based" et un PLNE) qui, contrairement à la littérature existante, sont capables de s'adapter à des entrepôts génériques avec un nombre arbitraire de blocs. La méthode de décomposition a été enrichie par un ensemble de techniques d'accélération qui ont grandement contribué au succès de la méthode et a sa supériorité par rapport au MILP.

Enfin, nous sommes restés dans le cadre des entrepôts à étagère mixtes et nous avons étudié un problème plus générique dans lequel une vague de demandes devait être récupérée dans la prochaine tournée d'un ensemble de préparateurs, chacun opérant dans une zone donnée (entrepôt découpé en zones). Nous avons proposé des adaptations de méthodes, initialement

développées pour le PRP-MS, pour résoudre ce nouveau problème. Nous avons aussi étudié une question pratique, à savoir l'impact du passage d'un stockage traditionnel à un stockage à étagères mixtes. Les résultats ont démontré un impact positif avec d'importants gains enregistrés pour le temps de collecte.

En conclusion, nous avons proposé dans le cadre de cette thèse un ensemble des méthodes exactes et approchées pour des problèmes de planification dans des entrepôts de commerce électronique. Ces méthodes sont capables de fournir des solutions de bonne qualité en des temps raisonnables, et peuvent donc être exploitées en pratique par des entrepôts faisant face aux mêmes défis que ceux considérés dans cette thèse. De plus, elles peuvent être utilisées comme des modules dans une méthode plus générique intégrant d'autres niveaux de décision ou d'autres pratiques émergentes dans le commerce électronique.

RÉFÉRENCES

[1] J. A. Tompkins, J. A. White, Y. A. Bozer et J. M. A. Tanchoco, *Facilities planning*. John Wiley & Sons, 2010.

[2] N. Boysen, K. Stephan et F. Weidinger, "Manual order consolidation with put walls : the batched order bin sequencing problem," *EURO Journal on Transportation and Logistics*, vol. 8, n°. 2, p. 169–193, 2019.

[3] J. Totonchi et G. Kakamanshadi, "Globalization and e-commerce," dans *2nd International Conference on Networking and Information Technology*, 2011, p. 270–276.

[4] Statista, *Retail e-commerce revenue in Canada from 2017 to 2025*, accessed on September 16, 2021, https://www.statista.com/statistics/289741/canada-retail-e-commerce-sales.

[5] F. Weidinger, N. Boysen et M. Schneider, "Picker routing in the mixed-shelves warehouses of e-commerce retailers," *European Journal of Operational Research*, vol. 274, n°. 2, p. 501–515, 2019.

[6] J. Gu, M. Goetschalckx et L. F. McGinnis, "Research on warehouse operation : A comprehensive review," *European Journal of Operational Research*, vol. 177, n°. 1, p. 1–21, 2007.

[7] R. B. De Koster, T. Le-Duc et N. Zaerpour, "Determining the number of zones in a pick-and-sort order picking system," *International Journal of Production Research*, vol. 50, n°. 3, p. 757–771, 2012.

[8] T.-L. Chen, C.-Y. Cheng, Y.-Y. Chen et L.-K. Chan, "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem," *International Journal of Production Economics*, vol. 159, p. 158–167, 2015.

[9] T. Van Gils, K. Ramaekers, A. Caris et R. B. de Koster, "Designing efficient order picking systems by combining planning problems : State-of-the-art classification and review," *European Journal of Operational Research*, vol. 267, n°. 1, p. 1–15, 2018.

[10] M. Schiffer, N. Boysen, P. S. Klein, G. Laporte et M. Pavone, "Optimal picking policies in e-commerce warehouses," *Management Science*, 2022.

[11] K. J. Roodbergen et R. De Koster, "Routing order pickers in a warehouse with a middle aisle," *European Journal of Operational Research*, vol. 133, n°. 1, p. 32–43, 2001.

[12] J. Gu, M. Goetschalckx et L. F. McGinnis, "Solving the forward-reserve allocation problem in warehouse order picking systems," *Journal of the Operational Research Society*, vol. 61, n°. 6, p. 1013–1021, 2010.

[13] R. De Koster, T. Le-Duc et K. J. Roodbergen, "Design and control of warehouse order picking : A literature review," *European Journal of Operational Research*, vol. 182, n°. 2, p. 481–501, 2007.

[14] Z. Chen et Y. Li, "Dual-command operation generation in bi-directional flow-rack automated storage and retrieval systems with random storage policy," dans *Proceedings of the 2015 Chinese Intelligent Systems Conference*. Springer, 2016, p. 125–131.

[15] J. L. Heskett, "Cube-per-order index-a key to warehouse stock location," *Transportation and distribution Management*, vol. 3, n°. 1, p. 27–31, 1963.

[16] C. G. Petersen, G. R. Aase et D. R. Heiser, "Improving order-picking performance through the implementation of class-based storage," *International Journal of Physical Distribution & Logistics Management*, 2004.

[17] F. Weidinger et N. Boysen, "Scattered storage : how to distribute stock keeping units all around a mixed-shelves warehouse," *Transportation Science*, vol. 52, n°. 6, p. 1412–1427, 2018.

[18] H. M. G. Albán, T. Cornelissens et K. Sörensen, *Scattered storage assignment : mathematical model and valid inequalities to optimize the intra-order item distances*. University of Antwerp, Faculty of Business and Economics, 2020.

[19] W. Jiang, J. Liu, Y. Dong et L. Wang, "Assignment of duplicate storage locations in distribution centres to minimise walking distance in order picking," *International Journal of Production Research*, vol. 59, n°. 15, p. 4457–4471, 2021.

[20] P. J. Parikh et R. D. Meller, "Selecting between batch and zone order picking strategies in a distribution center," *Transportation Research Part E : Logistics and Transportation Review*, vol. 44, n°. 5, p. 696–719, 2008.

[21] C. G. Petersen, "Considerations in order picking zone configuration," *International Journal of Operations & Production Management*, 2002.

[22] N. Boysen, R. de Koster et F. Weidinger, "Warehousing in the e-commerce era : A survey," *European Journal of Operational Research*, 2018.

[23] S. Saylam, M. Çelik et H. Süral, "The min–max order picking problem in synchronised dynamic zone-picking systems," *International Journal of Production Research*, p. 1–18, 2022.

[24] R. de Koster, "Performance approximation of pick-to-belt orderpicking systems," *European Journal of Operational Research*, vol. 72, n°. 3, p. 558–573, 1994.

[25] M. Yu et R. B. De Koster, "The impact of order batching and picking area zoning on order picking system performance," *European Journal of Operational Research*, vol. 198, n°. 2, p. 480–490, 2009.

[26] D. Roy, S. Nigam, R. de Koster, I. Adan et J. Resing, "Robot-storage zone assignment strategies in mobile fulfillment systems," *Transportation Research Part E : Logistics and Transportation Review*, vol. 122, p. 119–142, 2019.

[27] H. D. Ratliff et A. S. Rosenthal, "Order-picking in a rectangular warehouse : a solvable case of the traveling salesman problem," *Operations Research*, vol. 31, n°. 3, p. 507–521, 1983.

[28] G. Gutin et A. P. Punnen, *The traveling salesman problem and its variations*. Springer Science & Business Media, 2006, vol. 12.

[29] L. Pansart, N. Catusse et H. Cambazard, "Exact algorithms for the order picking problem," *Computers & Operations Research*, vol. 100, p. 117–127, 2018.

[30] A. N. Letchford, S. D. Nasiri et D. O. Theis, "Compact formulations of the steiner traveling salesman problem and related problems," *European Journal of Operational Research*, vol. 228, n°. 1, p. 83–92, 2013.

[31] H. Cambazard et N. Catusse, "Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane," *European Journal of Operational Research*, vol. 270, n°. 2, p. 419–429, 2018.

[32] C. Theys, O. Bräysy, W. Dullaert et B. Raa, "Using a tsp heuristic for routing order pickers in warehouses," *European Journal of Operational Research*, vol. 200, n°. 3, p. 755–763, 2010.

[33] A. Miller, "Order picking for the 21st century," *Manufacturing & Logistics IT*, 2004.

[34] F. Weidinger, "Picker routing in rectangular mixed shelves warehouses," *Computers & Operations Research*, vol. 95, p. 139–150, 2018.

[35] D. Goeke et M. Schneider, "Modeling single-picker routing problems in classical and modern warehouses," *INFORMS Journal on Computing*, vol. 33, n°. 2, p. 436–451, 2021.

[36] G. Wäscher, "Order picking : a survey of planning problems and methods," dans *Supply chain management and reverse logistics*. Springer, 2004, p. 323–347.

[37] N. Gademann et S. Velde, "Order batching to minimize total travel time in a parallel-aisle warehouse," *IIE transactions*, vol. 37, n°. 1, p. 63–75, 2005.

[38] T. Öncan, "Milp formulations and an iterated local search algorithm with tabu thresholding for the order batching problem," *European Journal of Operational Research*, vol. 243, n°. 1, p. 142–155, 2015.

[39] Y. A. Bozer et J. W. Kile, "Order batching in walk-and-pick order picking systems," *International Journal of Production Research*, vol. 46, n°. 7, p. 1887–1909, 2008.

[40] D. R. Gibson et G. P. Sharp, “Order batching procedures,” *European Journal of Operational Research*, vol. 58, n°. 1, p. 57–67, 1992.

[41] E. A. Elsayed, “Algorithms for optimal material handling in automatic warehousing systems,” *The International Journal of Production Research*, vol. 19, n°. 5, p. 525–535, 1981.

[42] G. Clarke et J. W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, vol. 12, n°. 4, p. 568–581, 1964.

[43] M. Albareda-Sambola, A. Alonso-Ayuso, E. Molina et C. S. De Blas, “Variable neighborhood search for order batching in a warehouse,” *Asia-Pacific Journal of Operational Research*, vol. 26, n°. 05, p. 655–683, 2009.

[44] S. Henn, S. Koch, K. F. Doerner, C. Strauss et G. Wäscher, “Metaheuristics for the order batching problem in manual order picking systems,” *Business Research*, vol. 3, n°. 1, p. 82–105, 2010.

[45] S. Henn et G. Wäscher, “Tabu search heuristics for the order batching problem in manual order picking systems,” *European Journal of Operational Research*, vol. 222, n°. 3, p. 484–494, 2012.

[46] S. Henn, S. Koch et G. Wäscher, “Order batching in order picking warehouses : a survey of solution approaches,” dans *Warehousing in the Global Supply Chain*. Springer, 2012, p. 105–137.

[47] O. Kulak, Y. Sahin et M. E. Taner, “Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms,” *Flexible Services and Manufacturing Journal*, vol. 24, n°. 1, p. 52–80, 2012.

[48] C. A. Valle, J. E. Beasley et A. S. da Cunha, “Optimally solving the joint order batching and picker routing problem,” *European Journal of Operational Research*, vol. 262, n°. 3, p. 817–834, 2017. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221717303004

[49] C. A. Valle et J. E. Beasley, “Order batching using an approximation for the distance travelled by pickers,” *European Journal of Operational Research*, vol. 284, n°. 2, p. 460–484, 2020.

[50] A. Scholz et G. Wäscher, “Order batching and picker routing in manual order picking systems : the benefits of integrated routing,” *Central European Journal of Operations Research*, vol. 25, n°. 2, p. 491–520, 2017.

[51] M. Bué, D. Cattaruzza, M. Ogier et F. Semet, “A two-phase approach for an integrated order batching and picker routing problem,” dans *A View of Operations Research Applications in Italy, 2018*. Springer, 2019, p. 3–18.

[52] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier et M. Ogier, "An efficient and general approach for the joint order batching and picker routing problem," *European Journal of Operational Research*, vol. 285, n°. 2, p. 497 – 512, 2020. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0377221720300977

[53] F. Weidinger, N. Boysen et M. Schneider, "Picker routing in the mixed-shelves warehouses of e-commerce retailers," *European Journal of Operational Research*, vol. 274, n°. 2, p. 501–515, 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221718308749

[54] P. Yang, Z. Zhao et H. Guo, "Order batch picking optimization under different storage scenarios for e-commerce warehouses," *Transportation Research Part E : Logistics and Transportation Review*, vol. 136, p. 101897, 2020.

[55] E. Elsayed, M.-K. Lee, S. Kim et E. Scherer, "Sequencing and batching procedures for minimizing earliness and tardiness penalty of order retrievals," *The International Journal of Production Research*, vol. 31, n°. 3, p. 727–738, 1993.

[56] S. Henn et V. Schmid, "Metaheuristics for order batching and sequencing in manual order picking systems," *Computers & Industrial Engineering*, vol. 66, n°. 2, p. 338–351, 2013.

[57] S. Henn, "Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses," *Flexible Services and Manufacturing Journal*, vol. 27, n°. 1, p. 86–114, 2015.

[58] A. Scholz, D. Schubert et G. Wäscher, "Order picking with multiple pickers and due dates–simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems," *European Journal of Operational Research*, vol. 263, n°. 2, p. 461 – 478, 2017. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0377221717303855

[59] C.-Y. Tsai, J. J. H. Liou et T.-M. Huang, "Using a multiple-ga method to solve the batch picking problem : considering travel distance and order due time," *International Journal of Production Research*, vol. 46, n°. 22, p. 6533–6555, 2008. [En ligne]. Disponible : https://doi.org/10.1080/00207540701441947

[60] T. Van Gils, A. Caris, K. Ramaekers et K. Braekers, "Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse," *European Journal of Operational Research*, vol. 277, n°. 3, p. 814–830, 2019.

[61] E. Ardjmand, H. Shakeri, M. Singh et O. Sanei Bajgiran, "Minimizing order picking makespan with multiple pickers in a wave picking warehouse," *International*

Journal of Production Economics, vol. 206, p. 169–183, 2018. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0925527318304146

[62] S. A. B. Rasmi, Y. Wang et H. Charkhgard, "Wave order picking under the mixed-shelves storage strategy : A solution method and advantages," *Computers & Operations Research*, vol. 137, p. 105556, 2022.

[63] W. Gu, D. Cattaruzza, M. Ogier et F. Semet, "Adaptive large neighborhood search for the commodity constrained split delivery vrp," *Computers & Operations Research*, vol. 112, p. 104761, 2019.

[64] J. Drury, "Towards more efficient order picking," *IMM monograph*, vol. 1, 1988.

[65] Statista, *Retail e-commerce revenue in Canada from 2017 to 2023*, accessed on December 2, 2019, https://www.statista.com/statistics/289741/canada-retail-e-commerce-sales.

[66] N. Boysen, R. de Koster et F. Weidinger, "Warehousing in the e-commerce era : A survey," *European Journal of Operational Research*, vol. 277, n°. 2, p. 396–411, 2019.

[67] C. A. Valle, J. E. Beasley et A. S. da Cunha, "Optimally solving the joint order batching and picker routing problem," *European Journal of Operational Research*, vol. 262, n°. 3, p. 817–834, 2017.

[68] N. Boysen, S. Fedtke et F. Weidinger, "Optimizing automated sorting in warehouses : The minimum order spread sequencing problem," *European Journal of Operational Research*, vol. 270, n°. 1, p. 386–400, 2018.

[69] T. Le-Duc et R. B. De Koster, "Travel distance estimation and storage zone optimization in a 2-block class-based storage strategy warehouse," *International Journal of Production Research*, vol. 43, n°. 17, p. 3561–3581, 2005.

[70] M. Masae, C. H. Glock et P. Vichitkunakorn, "Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour," *International Journal of Production Research*, vol. 58, n°. 17, p. 5337–5358, 2020.

[71] Ö. Öztürkoğlu et D. Hoser, "A discrete cross aisle design model for order-picking warehouses," *European Journal of Operational Research*, vol. 275, n°. 2, p. 411–430, 2019.

[72] M. Masae, C. H. Glock et P. Vichitkunakorn, "Optimal order picker routing in the chevron warehouse," *IISE Transactions*, vol. 52, n°. 6, p. 665–687, 2020.

[73] Ö. Öztürkoğlu, K. R. Gue et R. D. Meller, "Optimal unit-load warehouse designs for single-command operations," *IIE Transactions*, vol. 44, n°. 6, p. 459–475, 2012.

[74] M. Löffler, N. Boysen et M. Schneider, "Picker routing in agv-assisted order picking systems," *INFORMS Journal on Computing*, vol. 34, n°. 1, p. 440–462, 2022.

[75] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, n°. 12, p. 1985–2002, 2004.

[76] K.-J. Roodbergen, "Layout and routing methods for warehouses," Thèse de doctorat, Erasmus University Rotterdam, may 2001. [En ligne]. Disponible : http://hdl.handle.net/1765/861

[77] P. Laborie, "An update on the comparison of MIP, CP and hybrid approaches for mixed resource allocation and scheduling," dans *International conference on the integration of constraint programming, artificial intelligence, and operations research*. Springer, 2018, p. 403–411.

[78] A. M. Ham et E. Cakici, "Flexible job shop scheduling problem with parallel batch processing machines : Mip and cp approaches," *Computers & Industrial Engineering*, vol. 102, p. 160–165, 2016.

[79] P. Baptiste, C. Le Pape et W. Nuijten, *Constraint-based scheduling : applying constraint programming to scheduling problems*. Springer Science & Business Media, 2012, vol. 39.

[80] J. N. Hooker et W.-J. van Hoeve, "Constraint programming and operations research," *Constraints*, vol. 23, n°. 2, p. 172–195, 2018.

[81] P. Laborie, J. Rogerie, P. Shaw et P. Vilím, "IBM ILOG CP optimizer for scheduling," *Constraints*, vol. 23, n°. 2, p. 210–250, 2018.

[82] A. Bhatti, H. Akram, H. M. Basit, A. U. Khan, S. M. Raza et M. B. Naqvi, "E-commerce trends during covid-19 pandemic," *International Journal of Future Generation Communication and Networking*, vol. 13, n°. 2, p. 1449–1452, 2020.

[83] M. Masae, C. H. Glock et E. H. Grosse, "Order picker routing in warehouses : A systematic literature review," *International Journal of Production Economics*, vol. 224, p. 107564, 2020.

[84] R. Rahmaniani, T. G. Crainic, M. Gendreau et W. Rei, "The Benders decomposition algorithm : A literature review," *European Journal of Operational Research*, vol. 259, n°. 3, p. 801–817, 2017.

[85] S. Emde, L. Polten et M. Gendreau, "Logic-based Benders decomposition for scheduling a batching machine," *Computers & Operations Research*, vol. 113, p. 104777, 2020.

[86] A. Riise, C. Mannino et L. Lamorgese, "Recursive logic-based benders' decomposition for multi-mode outpatient scheduling," *European Journal of Operational Research*, vol. 255, n°. 3, p. 719–728, 2016.

[87] R. F. Fachini et V. A. Armentano, "Logic-based Benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows," *Computers & Industrial Engineering*, vol. 148, p. 106641, 2020.

[88] D. Wheatley, F. Gzara et E. Jewkes, “Logic-based Benders decomposition for an inventory-location problem with service constraints,” *Omega*, vol. 55, p. 10–23, 2015.

[89] Z. Zhang, X. Song, H. Huang, X. Zhou et Y. Yin, “Logic-based Benders decomposition method for the seru scheduling problem with sequence-dependent setup time and dejong’s learning effect,” *European Journal of Operational Research*, vol. 297, n°. 3, p. 866–877, 2022.

[90] C. G. Petersen, X. Cao et G. R. Aase, “Covid-19 pandemic implications for order picking operations,” *Open Journal of Business and Management*, vol. 9, n°. 6, p. 2866–2878, 2021.

[91] M. Masae, C. H. Glock et P. Vichitkunakorn, “A method for efficiently routing order pickers in the leaf warehouse,” *International Journal of Production Economics*, vol. 234, p. 108069, 2021. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0925527321000451

[92] M. Haouassi, Y. Kergosien, J. Mendoza et L.-M. Rousseau, “The picker routing problem in mixed-shelves multi-block warehouses,” Working Paper, 2022.

[93] M. Desrochers et G. Laporte, “Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints,” *Operations Research Letters*, vol. 10, n°. 1, p. 27–36, 1991.

ANNEXE A SUPPLEMENTS OF CHAPTER 3

A.1 Mathematical formulation

We propose a mixed-integer linear formulation (MILP) to model the problem. The formulation is defined on a new directed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$. The node set $\mathcal{V}' = \{0, n+1\} \cup \mathcal{L}$ contains two copies of the depot ($\{0, n+1\}$) in addition to the pick locations set $\mathcal{L}$. The arc set $\mathcal{A}'$ is composed of arcs from 0 to the nodes in $\mathcal{V}' - \{0\}$, arcs from the nodes in $\mathcal{V}' - \{0, n+1\}$ to $n+1$, and two directed arcs between each pair of nodes in $\mathcal{V}' - \{0, n+1\}$. Arc $(0, n+1)$ models an empty tour with null processing time $t_{0,n+1} = 0$. Note that the travel time between pick location i and the two depot copies are equivalent ($t_{0,i} = t_{i,n+1}$). Moreover, the induced sub-graph $\mathcal{G}'[\mathcal{L}]$ is complete and symmetric ($t_{i,j} = t_{j,i} | \forall i, j \in \mathcal{L}$). Table A.1 presents the sets, parameters and variables used in the following MILP formulation.

Table A.1 Notations used in the MILP formulation

Sets	
$\mathcal{O}$	Set of customer orders, where $\mathcal{O} = \{1, \ldots, o, \ldots, O\}$
$\mathcal{K}$	Set of order pickers, where $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$
$\mathcal{P}$	Set of positions, where $\mathcal{P} = \{1, \ldots, p, \ldots, P\}$. Note that P represents the maximum number of tours that can be assigned to a picker.
$\mathcal{M}_o$	Set of orderlines of customer orders o
$\mathcal{L}$	Set of storage locations, where $\mathcal{L} = \{1, \ldots, i, \ldots, n\}$
$\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$	Auxiliary graph with $\mathcal{V}'$ the node set and $\mathcal{A}'$ the arc set
Parameters	
W	Capacity of the cart
q_m	Weight of orderline m
d_o	Deadline of customer order o
$t_{i,j}$	Travel time between nodes i and j
β_s	Setup time of each tour
β_p	Search and pick time of each orderline
l_m	Index location of orderline m
M	A sufficiently large positive number
Variables	
$x_{m,o}^{k,p} \in \{0,1\}$	Equals 1 if orderline m of order o is picked by the picker k in the p-th tour, 0 otherwise.
$y_{i,j}^{k,p} \in \{0,1\}$	Equals 1 if the picker k uses arc (i,j) in the p-th tour, 0 otherwise.
$et_o \in \mathbb{R}^+$	Completion time of order o.
$at_i^{k,p} \in \mathbb{R}^+$	Arrival time of picker k to node i in the p-th tour.

$$\min \sum_{k \in \mathcal{K}} at_{n+1}^{k,P} \tag{A.1}$$

$$\sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}} x_{m,o}^{k,p} = 1 \qquad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}_o \tag{A.2}$$

$$\sum_{o \in \mathcal{O}} \sum_{m \in \mathcal{M}_o} q_m \cdot x_{m,o}^{k,p} \leq W \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.3}$$

$$\sum_{i \in \mathcal{L} \cup \{0\}} y_{i,j}^{k,p} - \sum_{i \in \mathcal{L} \cup \{n+1\}} y_{j,i}^{k,p} = 0 \qquad \forall j \in \mathcal{L}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.4}$$

$$\sum_{i \in \mathcal{V}' - \{0\}} y_{0,i}^{k,p} = 1 \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.5}$$

$$\sum_{i \in \mathcal{V}' - \{n+1\}} y_{i,n+1}^{k,p} = 1 \qquad \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.6}$$

$$t_{i,j} - M(1 - y_{i,j}^{k,p}) \leq at_j^{k,p} - at_i^{k,p} \quad \forall (i,j) \in \mathcal{A}', \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.7}$$
$$+ (\beta^s)_{\mathbb{1}_{(i=0 \ \cap \ j \neq n+1)}}$$
$$+ \left(\beta^p \cdot \sum_{o \in \mathcal{O}} \sum_{m \in \mathcal{M}_o | i = l_m} x_{m,o}^{k,p} \right)_{\mathbb{1}_{(i \neq 0)}}$$

$$at_{n+1}^{k,p} \leq at_0^{k,p+1} \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P} - \{P\} \tag{A.8}$$

$$\sum_{j \in \mathcal{L}} y_{0,j}^{k,p} \geq \sum_{j \in \mathcal{L}} y_{0,j}^{k,p+1} \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P} - \{P\} \tag{A.9}$$

$$\sum_{j \in \mathcal{L} \cup \{n+1\}} y_{l_m,j}^{k,p} \geq x_{m,o}^{k,p} \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}_o, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.10}$$

$$et_o \geq at_{n+1}^{k,p} - M \cdot (1 - x_{m,o}^{k,p}) \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}_o, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{A.11}$$

$$et_o \leq d_o \quad \forall o \in \mathcal{O} \tag{A.12}$$

$$et_o \geq 0, \ at_i^{k,p} \in \{0,1\} \quad \forall o \in \mathcal{O}, \forall i \in \mathcal{V}', \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \tag{A.13}$$

$$x_{m,o}^{k,p} \in \{0,1\}, \ y_{i,j}^{k,p} \in \{0,1\} \quad \forall o \in \mathcal{O}, \forall m \in \mathcal{M}_o, \forall (i,j) \in \mathcal{A}' \quad \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \tag{A.14}$$

The objective function (A.1) is the sum of pickers completion time. Assuming that the horizon start time equals 0, it is equivalent to minimizing the total processing time defined in equation (4.1) since there is no waiting times between tours and at picking point in the optimal solution. Constraints (A.2) assign each orderline to exactly one position of one picker. Constraints (A.3) ensure that each tour satisfies the cart capacity. Constraints (A.4), (A.5),(A.6) are flow constraints for each picker's tour. Constraints (A.7) are an adaptation of the subtour elimination constraints of Miller-Tucker-Zemlin (see [93]) that use nodes arrival time variables. Besides the travel time between the nodes of arc (i,j) in the current tour (p,k), a setup time is added to the constraint if $(i = 0)$ and $(j \neq n+1)$. Furthermore, the search and pick time of all orderlines retrieved from the tail of the arc (i,j) are added to the constraint if $(i \neq 0)$. Constraints (A.8) prevent overlaps between the consecutive tours of each picker. Constraints (A.9) ensure that the non-empty tours of each picker are positioned at the begenning of the sequence to avoid the exploration of some symmetric solutions (*i.e.*, symmetries that result in having an empty tour at different positions between two non-empty tours). Constraints (A.10) link the variable $x_{m,o}^{k,p}$ and $y_{i,j}^{k,p}$: A picker k must stop at the picking locations of all orderlines that he/she retrieves during his/her tour at position p. Constraints (A.11) define the completion time of each order as the completion time of the last tour that retrieves one of its orderlines. Constraints (A.12) force the satisfaction of the deadlines. Finally, constraints (A.13) and (A.14) define the domain of the variables.

To test our MILP formulation, we conducted preliminary experiments on a benchmark of small instances generated by [60]. In these instances, the number of orders is set to $\{18, 12, 6\}$ and the batch capacity is set to $\{4, 8, 12\}$. We ran the MILP formulation on some of those instances by setting a time limit of 2 hours. We observed that the MILP is not able to return a feasible solution for most of the instances, even for the smallest ones (6 orders, $W = 15$). For the few instances where the MILP returned feasible solutions, the gaps were poor (no less than 100%). We thus conclude that those results are not exploitable in any comparative analysis.

A.2 Kruskall-Wallis test

Table A.2 presents results of a Kruskal-Wallis H test on order picking time, on CPU time, and on gap $\Delta_{ILS|RFSS^+}$. Tables A.3, A.4, and A.5 present results of a pairwise-comparison between groups of each warehouse factor on order picking time, on CPU time, and on gap $\Delta_{ILS|RFSS^+}$ using the test of Dunn. Note that the Test of Dunn is done when the $p - value$ returned by the Kruskall-Wallis' test is significant (*i.e.*, $p - value < 0.05$).

Table A.2 Kruskal-Wallis H test on order picking time, on CPU time, and on gap $\Delta_{ILS|RFSS^+}$

	N	Statistic	df	p-value
Kruskal-Wallis H test on order picking time				
Layout	7290	2340	2	0.000
Storage policy	7290	32.3	2	0.0000001
Cart capacity	7290	4110.	2	0
Order strcture	7290	318.	2	7.94e-70
Deadline distribution	7290	2.69	2	0.26
Kruskal-Wallis H test on CPU time				
Layout	7290	1607.	2	0
Storage policy	7290	2353.	2	0
Cart capacity	7290	1823.	2	0
Order strcture	7290	204.	2	5.47e-45
Deadline distribution	7290	54.5	2	1.47e-12
Kruskal-Wallis H test on $\Delta_{ILS\|RFSS^+}$				
Layout	7290	5129.	2	0
Storage policy	7290	543.	2	1.35e-118
Cart capacity	7290	320.	2	2.63e-70
Order strcture	7290	780.	2	4.37e-170
Deadline distribution	7290	19.2	2	0.000067

Table A.3 Dunn test on order picking time

	N_1	N_2	Statstic	p.adj	p.adj.signif
Layout					
SW - MW	2430	2430	-26.0	5.67e-149	$****$
MW - LW	2430	2430	22.3	1.50e-109	$****$
LW - SW	2430	2430	-48.3	0.	$****$
Storage policy					
AA - Ran	2430	2430	4.62	0.0000114	$****$
WA - AA	2430	2430	-0.558	1	ns
Ran - WA	2430	2430	-5.18	0.000000667	$****$
Cart capacity					
15 - 30	2430	2430	-39.9	0.	$****$
30 - 45	2430	2430	-23.5	9.20e-122	$****$
45 - 15	2430	2430	-63.4	0.	$****$
Order strcture					
100 - 200	2430	2430	11.2	1.74e-28	$****$
200 - 300	2430	2430	6.46	3.11e-10	$****$
300 - 100	2430	2430	17.6	4.32e-69	$****$

Table A.4 Dunn test on CPU time

	N_1	N_2	Statstic	p.adj	p.adj.signif
Layout					
SW - MW	2430	2430	-26.5	1.02e-153	****
MW - LW	2430	2430	12.9	2.14e-37	****
LW - SW	2430	2430	-39.3	0.	****
Storage policy					
AA - Ran	2430	2430	-37.7	0.	****
WA - AA	2430	2430	-45.3	0.	****
Ran - WA	2430	2430	-7.56	1.24e-13	****
Cart capacity					
15 - 30	2430	2430	27.9	1.06e-170	****
30 - 45	2430	2430	14.1	2.18e-44	****
45 - 15	2430	2430	41.9	0.	****
Order strcture					
100 - 200	2430	2430	8.38	1.58e-16	****
200 - 300	2430	2430	5.82	1.77e-8	****
300 - 100	2430	2430	14.2	2.74e-45	****
Deadline distribution					
Deg - Prog	2430	2430	-3.45	1.66e-3	**
Deg - Uni	2430	2430	-7.38	4.85e-13	****
Prog - Uni	2430	2430	-3.92	2.62e-4	***

Table A.5 Dunn test on gap $\Delta_{ILS|RFSS^+}$

	N_1	N_2	Statstic	p.adj	p.adj.signif
Layout					
SW - MW	2430	2430	-47.4	0.	$****$
MW - LW	2430	2430	22.8	8.65e-115	$****$
LW - SW	2430	2430	-70.2	0.	$****$
Storage policy					
AA - Ran	2430	2430	18.6	1.77e- 76	$****$
WA - AA	2430	2430	-2.90	1.10e- 2	$*$
Ran - WA	2430	2430	-21.5	8.56e-102	$****$
Cart capacity					
15 - 30	2430	2430	14.8	3.51e-49	$****$
30 - 45	2430	2430	1.29	5.88e-1	ns
45 - 15	2430	2430	16.1	6.69e-58	$****$
Order strcture					
100 - 200	2430	2430	-14.5	5.64e-47	$****$
200 - 300	2430	2430	-13.5	9.09e-41	$****$
300 - 100	2430	2430	-27.9	4.45e-171	$****$
Deadline distribution					
Deg - Prog	2430	2430	-4.30	0.0000521	$**$
Deg - Uni	2430	2430	-2.90	0.0111	$*$
Prog - Uni	2430	2430	1.39	0.491	ns

A.3 Constraint programming formulation

We propose a natural CP formulation to model the problem. The formulation uses the parameters and variables summarized in Table A.6.

Table A.6 Notations used in the CP formulation

Parameters	
v^+	Artificial tour with $p_{v^+} = 0$ and $\tilde{d}_{v^+} = \max_{v \in \Pi} \tilde{d}_v$
Variables	
pos_v^k	Position number of tour v if assigned to picker k, -1 otherwise.
st_t^k	Start time of tour at position t of picker k
st_v	Start time of tour v
p_t^k	Processing time of tour at position t of picker k

$$\sum_{k=1}^{K} (pos_v^k \geq 1) = 1, \qquad \forall v \in \Pi \tag{A.15}$$

$$\sum_{v \in \Pi} (pos_v^k = t) \leq 1, \qquad \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P\} \tag{A.16}$$

$$st_1^k = 0, \qquad \forall k \in \{1, \ldots, K\} \tag{A.17}$$

$$st_{t+1}^k = st_t^k + p_t^k, \qquad \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P-1\} \tag{A.18}$$

$$pos_v^k = t \implies p_t^k = p_v, \qquad \forall v \in \Pi, \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P\} \tag{A.19}$$

$$pos_v^k = t \implies st_t^k = st_v, \qquad \forall v \in \Pi, \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P\} \tag{A.20}$$

$$st_v + p_v \leq \tilde{d}_v \qquad \forall v \in \Pi \cup \{v^+\} \tag{A.21}$$

$$p_t^k, st_v, st_t^k \in \mathbb{R}^+, \qquad \forall v \in \Pi, \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P\} \tag{A.22}$$

$$pos_v^k \in \{1, \ldots, P\}, \qquad \forall k \in \{1, \ldots, K\}, \forall t \in \{1, \ldots, P\}$$

Constraints (A.15) guarantee that each tour is sequenced once and only once. Constraints (A.16) ensure that no more then one tour is sequenced at the t^{th} position of picker k. Constraints (A.17) and (A.18) sequence the tours assigned to each picker. Constraints (A.19) link the variables pos_v^k and p_t^k while constraints (A.20) synchronize st_v and st_t^k variables. Constraints (A.21) bound the end time of each tour by its deadline. Finally, constraints (A.22) define the domain of the variables.

ANNEXE B SUPPLEMENTS OF CHAPTER 4

B.1 Appendix. Proof of validity of cuts

Proposition 3. *Cut (5.8) excludes the current master problem solution if it is not globally optimal.*

Proof. Let us assume that for a given iteration i, the master problem returns the solution vector (x^i, θ^i) with $\bar{\mathcal{C}}^i$ being the selected configurations and $\bar{\mathcal{P}}^i$ the picking locations included in these configurations. Furthermore, let $\bar{\theta}^i$ be the cost of the optimal tour returned by the subproblem to visit the picking locations in $\bar{\mathcal{P}}^i$. Then, the cut $(5.8)^i$ is added to the master problem with $\bar{\theta} = \bar{\theta}^i$, $\bar{\mathcal{P}} = \bar{\mathcal{P}}^i$, and $\bar{\mathcal{C}} = \bar{\mathcal{C}}^i$.

Let us assume that at iteration $i+1$, the previous master solution is not excluded (*i.e.*, $x^{i+1} = x^i$, $\bar{\mathcal{P}}^{i+1} = \bar{\mathcal{P}}^i$, and $\bar{\mathcal{C}}^{i+1} = \bar{\mathcal{C}}^i$). Given constraint $(5.8)^i$, it necessary follows that $(1 - x^c_{k,l}) = 0$ for all $c \in \bar{\mathcal{C}}^{i+1}$ and hence $\theta^{i+1} \geq \bar{\theta}^i$. Since the master problem seeks to minimize the value of θ, it results that the unique configuration that may be possible is that $\theta^{i+1} = \bar{\theta}^i$ which means that (x^i, θ^{i+1}) is the optimal solution. □

Proposition 4. *Cut (5.8) does not remove any globally optimal solution.*

Proof. We prove proposition (4) by contradiction.
Let us assume that the master problem returns a globally feasible solution (x^*, θ^*) that does not satisfy a cut of type (5.8) generated at iteration i, with $\bar{\mathcal{C}}^*$ (resp.$\bar{\mathcal{P}}^*$) being the set of selected configurations (resp. picking locations). Furthermore, let $\bar{\theta}^*$ be the cost of the optimal tour that visits the picking locations in $\bar{\mathcal{P}}^*$. Since cut $(5.8)^i$ is not satisfied, we have:

$$\bar{\theta}^* < \bar{\theta}^i - \sum_{(c,k,l) \in \bar{\mathcal{C}}^i} \sum_{p \in \mathcal{P}^c_{k,l}} \left[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p} \right] \tag{B.1}$$

Let us define the 3 following sets:

— $\alpha_1 = \{\bar{\mathcal{P}}^* \setminus \bar{\mathcal{P}}^i\}$
— $\alpha_2 = \{\bar{\mathcal{P}}^* \cap \bar{\mathcal{P}}^i\}$
— $\alpha_3 = \{\bar{\mathcal{P}}^i \setminus \bar{\mathcal{P}}^*\}$

(B.1) can be rewritten as follows:

$$(B.1) \iff \bar{\theta}^* < \bar{\theta}^i - \sum_{(c,k,l)\in\bar{\mathcal{C}}^i} \sum_{p\in\mathcal{P}^c_{k,l}: p\in\alpha_2\cup\alpha_3} \Big[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p}\Big] \tag{B.2}$$

We distinguish two cases:

1. $\alpha_2 = \emptyset$. It means that the solution that violates the cut $(5.8)^i$ does not share any picking locations with the solution that yields to the generation of cut $(5.8)^i$ (*i.e.*, $\alpha_3 = \bar{\mathcal{P}}^i$). It follows that $(x^c_{k,l})^* = 0, \forall (c,k,l) \in \bar{\mathcal{C}}^i, \forall p \in \{\mathcal{P}^c_{k,l} : p \in \alpha_3\}$. Let $\pi^i = \{0, 1^i, 2^i, \cdots, p^i, p^{i+1}, \cdots, P^i, 0\}$ be the optimal path returned by the subproblem at iteration i. Due to the triangular inequality, we have:

$$\begin{aligned}
d_{0,1^i} &= d_{0,1^i} \\
d_{1^i,2^i} &\leq d_{0,1^i} + d_{0,2^i} \\
d_{2^i,3^i} &\leq d_{0,2^i} + d_{0,3^i} \\
&\vdots \\
d_{p^i,(p+1)^i} &\leq d_{0,p^i} + + d_{0,(p+1)^i} \\
&\vdots \\
d_{(P-1)^i,P^i} &\leq d_{0,(P-1)^i} + d_{0,(P)^i} \\
d_{0,P^i} &= d_{0,P^i}
\end{aligned}$$

   By summing the terms of each side of the inequalities, we find:

$$\bar{\theta}^i \leq 2 \cdot \sum_{p\in\alpha_2\cup\alpha_3} d_{0,p} \tag{B.3}$$

   From (B.2) and (B.3), we deduce that $\bar{\theta}^* < 0$, a contradiction with constraints (5.7) of the master problem.

2. $\alpha_2 \neq \emptyset$. Let π^* be the optimal tour that visits the picking locations in $\bar{\mathcal{P}}^*$. Let π_{α_2} be the route obtained by removing from π^* the picking locations in α_1. Due to the triangular inequality, it necessarily follows that:

$$f(\pi_{\alpha_2}) \leq \bar{\theta}^* < \bar{\theta}^i - \sum_{(k,l,c)\in\bar{\mathcal{C}}^i} \sum_{p\in\mathcal{P}^c_{k,l}: p\in\alpha_3} \Big[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p}\Big] \tag{B.4}$$

   Let $\pi_{\alpha_2\cup\alpha_3}$ be a new path constructed from the path π_{α_2} by inserting each picking location in α_3 between the depot and the first picking location of the current path. If

$(0, z, \cdots, 0)$ is the current path and $p \in \alpha_3$ is the next picking location to insert, then the path after the insertion of p is the following $(0, p, z, \cdots, 0)$. Note that the insertion of the picking locations $p \in \alpha_3$ can be done in any order. Let $ICost(p, z)$ be the cost of the insertion of p between 0 and z $ICost(p, z) = d_{0,p} + d_{p,z} - d_{0,z} \geq 0$. Assuming that $d_{p,z} - d_{0,z} \leq d_{0,p}$ (triangular inequality), it follows that $ICost(p, z) \leq 2 \cdot d_{0,p}$. By repeating the same process for each picking location $p \in \alpha_3$ and observing that $f(\pi_{\alpha_2 \cup \alpha_3}) \geq \bar{\theta}^i$ (where $\bar{\theta}^i$ is the cost of the optimal tour that visits the picking locations in $\alpha_2 \cup \alpha_3$), we obtain:

$$\bar{\theta}^i \leq f(\pi_{\alpha_2 \cup \alpha_3}) = f(\pi_{\alpha_2}) + \sum_{p \in \alpha_3} ICost(p, z) \leq f(\pi_{\alpha_2}) + \sum_{p \in \alpha_3} 2 \cdot d_{0,p}$$
$$\leq f(\pi_{\alpha_2}) + \sum_{(c,k,l) \in \bar{\mathcal{C}}^i} \sum_{p \in \mathcal{P}^c_{k,l} : p \in \alpha_3} \Big[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p} \Big] \tag{B.5}$$

It follows that:

$$f(\pi_{\alpha_2}) \geq \bar{\theta}^i - \sum_{(c,k,l) \in \bar{\mathcal{C}}^i} \sum_{p \in \mathcal{P}^c_{k,l} : p \in \alpha_3} \Big[(1 - x^c_{k,l}) \cdot 2 \cdot d_{0,p} \Big] \tag{B.6}$$

We finally observe that (B.4) and (B.6) are in contradiction.

□