Titre: Title:	Programmation mathématique pour l'allocation des commandes et des étagères de stockage dans un entrepôt semi-automatisé de type RMFS
Auteur: Author:	Siryne El Amrani
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	El Amrani, S. (2025). Programmation mathématique pour l'allocation des commandes et des étagères de stockage dans un entrepôt semi-automatisé de type RMFS [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/71829/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/71829/
Directeurs de recherche: Advisors:	Thibaut Vidal, & Jorge Mendoza Gimenez
Programme: Program:	Maîtrise recherche en mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Programmation mathématique pour l'allocation des commandes et des étagères de stockage dans un entrepôt semi-automatisé de type RMFS

SIRYNE EL AMRANI
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques

Décembre 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Programmation mathématique pour l'allocation des commandes et des étagères de stockage dans un entrepôt semi-automatisé de type RMFS

présenté par **Siryne EL AMRANI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel GENDREAU, président
Thibaut VIDAL, membre et directeur de recherche
Jorge E. MENDOZA, membre et codirecteur de recherche
Martin COUSINEAU, membre

REMERCIEMENTS

Ce mémoire marque l'aboutissement de mon cursus de maîtrise et est le fruit d'un travail de recherche qui a bénéficié de l'accompagnement et du soutien de plusieurs personnes que je souhaite remercier ici.

Je tiens en premier lieu à exprimer ma gratitude à mon directeur de recherche, Pr Thibaut Vidal, et mon codirecteur de recherche, Pr Jorge Mendoza, pour la confiance qu'ils m'ont accordée en me confiant ce sujet portant sur l'optimisation des systèmes RMFS. Je les remercie pour leur disponibilité, la rigueur de leur encadrement scientifique et leurs conseils toujours avisés qui ont permis d'orienter mes travaux, notamment lors des phases de modélisation mathématique. Je remercie également Jiaqi Liang pour son suivi durant la première partie de ce projet.

J'adresse également mes remerciements aux membres du jury, Pr Michel Gendreau et Pr Martin Cousineau, pour l'intérêt qu'ils ont porté à ce travail et pour avoir accepté de l'évaluer.

Je souhaite aussi remercier M. Serge Bisaillon pour son aide technique précieuse concernant la prise en main du framework de simulation, pour sa bienveillance et pour nos échanges constructifs.

Je souhaite exprimer ma reconnaissance envers l'environnement stimulant du laboratoire. Je pense particulièrement à mes collègues Awa et Youssouf : leur écoute et nos discussions techniques, bien que ponctuelles, m'ont offert un recul précieux sur l'architecture du framework et ont été une véritable source de motivation.

J'ai plus généralement beaucoup apprécié la qualité des échanges quotidiens au sein de l'espace de vie du laboratoire. Ce lieu de convivialité attirant des chercheurs de tous les étages a grandement facilité mon intégration. Je souhaite saluer et remercier Nagisa, Awa, Utsav, Ansgar, Pierre-Yves, Julien, Maria, Théo, Youssouf pour ces moments de détente partagés.

Je tiens à exprimer ma sincère gratitude à Khalid Laaziri. Au-delà de ses compétences techniques, je le remercie pour sa gentillesse et la bienveillance de son écoute. Sa présence et son soutien humain ont été des atouts précieux tout au long de ce projet.

Merci à mes proches pour leur soutien constant et leur accompagnement précieux tout au long de ces années d'études.

RÉSUMÉ

Le secteur de la vente au détail connaît une profonde transformation en lien avec une connectivité toujours plus accrue. Les magasins physiques font à présent de plus en plus place aux commerces en ligne qui sont populaires et nombreux. Si les consommateurs sont férus de ce type de commerces pratiques et faciles d'accès, ils n'en sont pas moins exigeants. Ce secteur étant très concurrentiel, la fidélité des clients pour une firme de e-commerce n'est jamais garantie et repose en majeure partie sur des promesses d'efficacité en termes de délais de livraison. S'approcher de l'idéal du *same-day delivery* d'une manière qui puisse rester suffisamment économique pour une entreprise nécessite des moyens logistiques fins et optimaux. C'est dans ce contexte qu'ont été développés les entrepôts semi-automatisés dont la robotisation permet justement une gestion adaptable et efficace des ressources.

Les entrepôts semi-automatisés les plus communément utilisés dans le domaine du commerce en ligne sont les Robot Mobile Fulfillment Systems (RMFS). Il s'agit de systèmes d'entreposage dont la robotisation consiste en la présence d'une flotte de robots capables de se glisser sous les étagères de stockage, les soulever et les déplacer. Ces robots ont pour fonction d'exécuter des tâches consistant par exemple à ranger les étagères au sein de la zone de stockage ou encore à les apporter aux stations de remplissage ou aux stations de travail au sein desquelles un humain dépose ou récupère certains items dans les étagères. Le fonctionnement de ce système repose ainsi sur de multiples décisions opérationnelles le rendant très complexe. Cette complexité est d'autant plus accrue par le contexte très compétitif décrit précédemment. Les firmes sont contraintes de considérer les commandes passées par les consommateurs presque instantanément. Ceci aboutit à la gestion des problèmes de décisions évoqués en temps réel.

Ce mémoire de maîtrise a pour objectif d'optimiser le nombre de commandes traitées au sein d'un tel entrepôt semi-automatisé, par l'intermédiaire de la résolution simultanée des problèmes de décision pour l'allocation des commandes et des étagères aux stations de travail, à l'aide de la programmation mathématique en nombres entiers. Deux modèles aux formulations différentes y sont comparés à la fois à un système prenant ces décisions séquentiellement et également entre eux. L'un est formulé de façon flexible, ne reliant pas les commandes et étagères entre elles, et a été développé dans le cadre de cette maîtrise de recherche. L'autre

est plus rigide, reliant items, étagères, commandes et stations entre elles, et a été tiré d'un article de la littérature. Cette comparaison a pour objectif de détecter les forces et les faiblesses de l'un et l'autre de ces modèles afin de proposer une conclusion sur la formulation la plus adaptée au problème, qui pourra ensuite être utilisée en tant que *baseline* pour de futures recherches sur ce sujet. Dans l'optique de respecter au mieux le caractère dynamique et incertain de l'environnement constitué par un entrepôt de type RMFS, les deux modèles sont implémentés au sein d'une simulation très réaliste. C'est un point de ce travail qui est important à la fois méthodologiquement et techniquement, puisque cette simulation est un projet open source très dense dans lequel l'implémentation d'un quelconque modèle peut s'avérer complexe.

Les expériences menées dans cette étude ont montré, d'une part, que la modélisation de la prise de décision de façon simultanée des problèmes d'allocation considérés permet un regroupement plus efficace des commandes et un meilleur choix des étagères les plus intéressantes, comparé à une prise de décision séquentielle. D'autre part, la modélisation la plus flexible, celle développée à l'occasion de ce projet, a montré un réel avantage par rapport à la seconde qui était plus rigide. Elle s'est en effet démarquée par une meilleure utilisation des ressources de cet environnement dynamique. Nous avons pu observer une meilleure gestion des commandes lorsqu'elles sont nombreuses et une meilleure utilisation des robots lorsque leur nombre est faible. Ces observations sont à mettre en lien avec une meilleure adaptabilité du modèle flexible à l'arrivée continue de commandes vers l'entrepôt, en remettant constamment en jeu les étagères choisies lors de prises de décision antérieures. Ceci est rendu possible par l'indisponibilité récurrente des robots au sein de l'entrepôt aux moments où les décisions sont prises. Le modèle flexible profite de ces instants pour réviser les solutions données en lien avec l'arrivée de nouvelles commandes, tandis que le second modèle garde les solutions figées dans le temps, malgré le caractère dynamique de l'environnement.

ABSTRACT

The retail sector is undergoing a profound transformation driven by ever-increasing connectivity. Physical stores are now increasingly giving way to online commerce, which is both popular and numerous. While consumers are keen on this type of convenient and easily accessible retail, they are also highly demanding. As this sector is highly competitive, customer loyalty to an e-commerce firm is never guaranteed and relies primarily on promises of efficiency in terms of delivery times. Achieving the ideal of *same-day delivery* in a way that remains sufficiently economical for a company requires sophisticated and optimized logistical means. It is in this context that semi automated warehouses have been developed, where robotization allows for precisely an adaptable and efficient management of resources.

The most commonly used semi automated warehouses in the e-commerce domain are the Robot Mobile Fulfillment Systems (RMFS). These are warehousing systems whose robotization consists of a fleet of robots capable of sliding underneath storage shelves, lifting them, and moving them. These robots are responsible for executing tasks such as organizing the shelves within the storage area or bringing them to replenishment stations or workstations where a human places or retrieves items from the shelves. The operation of this system thus relies on multiple operational decisions, making it highly complex. This complexity is further exacerbated by the highly competitive context described above. Firms are constrained to consider consumer orders, which are placed, almost instantaneously. This results in the management of the aforementioned decision problems in real-time.

The objective of this Master's thesis is to optimize the number of orders processed within such a semi automated warehouse by simultaneously addressing two decision problems—the allocation of orders and shelves to workstations—using integer mathematical programming. Two models with different formulations are compared against a system that makes these decisions sequentially, as well as against each other. One model is flexibly formulated, not linking orders and shelves together, and was developed within the scope of this research thesis. The other is more rigid, linking items, shelves, orders, and stations together, and was adapted from a literature article. This comparison aims to identify the strengths and weaknesses of each model in order to propose a conclusion on the most suitable formulation

for our problem, which can then be used as a baseline for future research on this topic. In the interest of best respecting the dynamic and uncertain nature of the RMFS-type warehouse environment, both models in this work are implemented within a highly realistic simulation. This is an important aspect of this work, both methodologically and technically, as this simulation is a dense, open-source project in which implementing any model can prove complex.

The experiments conducted in this study showed, on the one hand, that modeling the decision-making process for the considered allocation problems simultaneously allows for more efficient order grouping and a better selection of the most suitable shelves, compared to sequential decision-making. On the other hand, the more flexible modeling approach, the one developed for this project, showed a real advantage over the second, more rigid model. It distinguished itself by a better utilization of resources in this dynamic environment. We observed improved order management when the number of orders is high and better robot utilization when their number is low. These observations are linked to the flexible model's superior adaptability to the continuous arrival of orders into the warehouse by constantly re-evaluating the shelves chosen during previous decisions. This is made possible by the recurring unavailability of robots within the warehouse at the moments when decisions are being made. The flexible model takes advantage of these moments to revise the proposed solutions based on the arrival of new orders, whereas the second model keeps the solutions fixed over time, despite the dynamic nature of the environment.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES DÉFINITIONS

Affectation des commandes de prélèvement

Ou POA: Allocation des commandes se trouvant dans le backlog aux différentes stations de travail du RMFS.

Agent

Au sein de la simulation RawSim-O, cela peut correspondre à des éléments réels tels que des robots ou des éléments virtuels propres à la simulation tels que les managers.

Allocation d'une tâche

Attribution d'un pod et d'une destination à un robot.

Backlog

Les commandes de prélèvement en attente dans le RMFS.

Balanced

Un contrôleur d'allocation des tâches visant simplement à équilibrer les robots entre les différentes stations.

Batching

Affectation de commandes de prélèvement en lots aux stations de travail du RMFS.

Brute Force

Une des meilleures règles d'allocation de tâche. Il s'agit de décider de la prochaine tâche pour chaque robot en vérifiant toutes les tâches possibles afin de trouver celle nécessitant le temps minimal pour stocker l'étagère actuelle, récupérer la suivante et l'amener à la station de destination de la tâche.

Business-to-consumer

Les transactions commerciales entre une entreprise et des clients particuliers.

Case

Boîte remplie par un travailleur au niveau des stations de travail et qui accueillera les items d'une commande particulière.

Class PSA

Une des meilleures règles PSA dans RawSim-O qui ramène les pods vers des emplacements de stockage de la même classe, mais en fonction du temps de trajet le plus court

vers une station de prélèvement. Au sein d'une classe, l'emplacement de stockage d'un pod est sélectionné de manière analogue à la règle du Nearest PSA (voir définition).

Class RPS

Ou Turnover Item Storage: Cette règle attribue les commandes de réapprovisionnement entrantes à un pod de la même classe que la commande de réapprovisionnement, c'est-à-dire les SKU à rotation rapide aux pods avec d'autres SKU à rotation rapide.

Commande de prélèvement

Ou pick order: Commande constituée de plusieurs lignes de SKU passée par un client et reçue par le RMFS.

Demand

L'une des meilleures règles PPS dans RawSim-O qui sélectionne l'étagère dont le contenu est le plus demandé compte tenu de la situation actuelle du backlog des commandes de prélèvement, c'est-à-dire que l'étagère contenant le plus d'unités demandées dans le backlog est choisie. Les égalités sont départagées aléatoirement.

Emptiest

Meilleure règle RPS dans RawSim-O qui attribue les commandes de réapprovisionnement à l'étagère la plus vide et réutilise la même étagère pour les commandes de réapprovisionnement suivantes jusqu'à ce qu'elle soit pleine ou utilisée dans une station.

Item

Une unité d'un SKU.

Nearest PPS

L'une des meilleures règles PPS dans RawSim-O qui sélectionne le pod qui a le temps de trajet estimé le plus court vers la station selon l'algorithme de planification de trajet et qui offre au moins une unité utile pour la sélection.

Nearest PSA

Une des meilleures règles PSA dans RawSim-O qui stocke les pods à l'emplacement de stockage inoccupé le plus proche, en fonction du temps de trajet estimé le plus court. Ce temps de trajet est déterminé à l'aide d'un algorithme A* qui prend en compte le temps nécessaire au retour du robot (avec ou sans pod).

Part to picker

Entrepôt semi-automatisé où des robots apportent aux travailleurs humains les items à prélever pour compléter les commandes.

Path Planning (PP)

Choix d'un chemin pour un pod en mouvement.

Pick Order Assignment (POA)

Problème de décision qui consiste à allouer les commandes de prélèvement du backlog vers une stations de travail.

Pick Pod Selection (PPS)

Problème de décision, lorsqu'un robot travaillant pour une station de prélèvement demande une tâche d'extraction suivante, un pod adapté au prélèvement à la station de prélèvement doit être sélectionné.

Picker to part

Entrepôt classique dans lequel les travailleurs humains se déplacent dans la zone de stockage pour récupérer les items pour compléter les commandes.

Pod

Etagères mobiles situées dans la zone de stockage de l'entrepôt sur lesquelles sont stockés les SKU.

Pod Batch

Meilleure règle ROA dans RawSim-O qui essaie d'utiliser un pod déjà sélectionné pour aller à une station de réapprovisionnement pour attribuer la prochaine commande de réapprovisionnement.

Pod Match

Meilleure règle POA dans RawSim-O qui sélectionne la commande de prélèvement dans le backlog qui correspond le mieux aux pods se dirigeant vers la ou les stations au moment de l'affectation.

Pod Repositioning (PR)

Autre désignation de Pod Storage Assignment.

Pod Storage Assignment (PSA)

Problème de décision, pour chaque pod, un emplacement de stockage inoccupé doit être sélectionné, à chaque fois, après avoir visité un poste de prélèvement ou de réapprovisionnement.

Problème de décision

Décisions opérationnelles prises dans le but de compléter les commandes de façon optimale. Cela inclut les problèmes POA, TA, PPS, PP, PR....

Préparation des commandes

Processus constitué de multiple décisions prises pour préparer les commandes reçues, en particulier l'affectation des commandes de prélèvement et l'affectation des pods aux stations de travail.

RawSim-O

Framework open source codé en `C#` et simulant un entrepôt RMFS permettant la prise de décisions en temps réel.

Replenishment Order Assignment (ROA)

Problème de décision, même principe que pour Pick Order Assignment, mais pour une commande de réapprovisionnement d'un pod.

Replenishment Pod Selection (RPS)

Problème de décision, même principe que pour Pick Pod Selection, mais pour remplir un pod aux stations de réapprovisionnement.

Robot

Robot passant sous un pod, le soulèvant et le transportant jusqu'à une station de travail.

Robotic Mobile Fulfillment System (RMFS)

Entrepôt semi-automatisé utilisé dans le cadre du e-commerce mettant en jeu l'utilisation de robots, de pod et de stations de travail.

Règle de décision

Différentes déclinaisons d'un problème de décision particulier.

Station Based

Une des meilleures règles PSA dans RawSim-O qui est une variante de la règle Nearest PSA: au lieu d'amener la capsule à l'emplacement de stockage le plus proche de la position du robot, l'emplacement de stockage offrant le temps de trajet le plus court vers une station de prélèvement est sélectionné. La principale différence avec la règle du plus proche réside dans les emplacements de stockage choisis pour les capsules revenant d'une visite à une station de réapprovisionnement.

Station de travail

Station au niveau desquelles les travailleurs prélèvent les items nécessaires d'un pod afin de compléter une commande avant de les placer dans une case.

Task Allocation (TA)

Problème de décision, attribution d'une tâche à un robot (très lié aux règles PPS et RPS).

Throughput rate

Moyenne du nombre de commandes effectuées par unité de temps.

Throughput time

Temps moyen mis pour traiter une commande ou pour prélever un item (unit ou order throughput time).

Trigger

Situation notifiée par l'un des managers du RMFS du framework RawSim-O, mis-à-jour par le simulateur qui va déclencher le contrôleur associé pour prendre une décision pour le problème de décision lié.

Turnover time

Délai moyen entre la soumission d'une commande de préparation au backlog et son exécution.

Tâche d'extraction

Ou picking: Dans le framework RawSim-O, SKU à prélever d'un pod pour compléter une commande de prélèvement .

Zone de stockage

Partie centrale de l'entrepôt où sont stockés les pods.

LISTE DES SIGLES ET ABRÉVIATIONS

RMFS	Robotic Mobile Fulfillment System
SKU	Stock Keeping Unit
POA	Pick Order Assignment
TA	Task Allocation
PPS	Pick Pod Selection
ROA	Replenishment Order Assignment
RPS	Replenishment Pod Selection
PSA	Pod Storage Assignment
PP	Path Planning
PR	Pod Repositioning
OP	Orders Placed
OH	Orders Handled
OTHT	Orders throughput time
OTOT	Orders turnover time
DT	Distance Traveled
IPO	Item pile-on
B2C	business-to-consumer
AS/RS	Automated Storage and Retrieval System
AVS/RS	Autonomous Vehicle Storage and Retrieval System

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Ce mémoire de recherche porte sur l'optimisation en temps réel d'une partie de la chaîne d'approvisionnement au sein des entrepôts semi-automatisés de type RMFS. Ces entrepôts se sont développés en lien avec le développement du commerce en ligne, un secteur très concurrentiel et qui s'accompagne d'exigences croissantes en termes de rapidité et d'efficacité. Cela s'est traduit par le développement d'un nouveau type de robotisation. Une flotte de robots *Amazon Robotics*, que l'on peut voir sur la FIGURE 1.1, a pour mission de se déplacer au sein de la zone de stockage de l'entrepôt sous les étagères afin de les soulever et d'assurer leur transport vers des opérateurs humains postés à des stations de travail pour traiter les commandes.

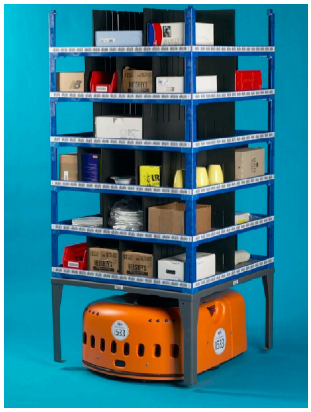

FIGURE 1.1 Photographie d'un robot sous une étagère de stockage tirée de l'article de Enright et al. [7]

Au niveau de ces stations sont exécutées et assemblées les diverses commandes reçues, dans des case prévues à cet effet, comme nous pouvons le voir sur la FIGURE 1.2.

FIGURE 1.2 Photographie d'une station de travail présentant un travailleur complétant des commandes à l'aide de les stocks apportés par un robot, tirée de l'article de Wurman et al. [20]

Une bonne gestion de ce système est nécessaire pour l'exploiter efficacement. En effet, "le problème de la coordination des robots dans les entrepôts rassemble des problèmes de calcul pour une variété de disciplines, notamment la planification, la prise de décision dans des conditions d'incertitude, l'exploration de données, l'apprentissage, la planification des trajectoires des robots et l'optimisation classique" (Enright et al. [7]). Parmi les nombreux problèmes de décisions, celles correspondant à l'allocation des commandes en lots et des étagères vers les stations de travail présentent une grande opportunité de bénéfices puisque regrouper les commandes de manière intelligente et choisir les étagères les plus adaptées pour les servir permettra une meilleure continuité au niveau des stations de travail, et donc un traitement plus efficace des requêtes des consommateurs.

1.1 Définitions et concepts de base

Le commerce en ligne a connu une croissance rapide au cours de la dernière décennie, celle-ci s'est maintenue à un niveau de 15% par an sur le marché américain. En effet, selon l'article de Cezik et al. [5], les achats en ligne permettent aux clients de passer commande à tout moment, offrent une large variété de produits et facilitent la comparaison entre différentes options. Selon le site internet Bureau du recensement des États-Unis [17], les ventes totales du commerce électronique dans le commerce de détail en 2020 se sont élevées à 815,4 milliards de dollars, représentant 14,6 % des ventes au détail. À titre de comparaison, selon cette même source, les ventes dans le commerce de détail du commerce électronique étaient

de 571 milliards de dollars en 2019, soit 10,6 % des ventes au détail. Selon l'ouvrage de Bartholdi et al. [2] : "tous les entrepôts sont optimisés pour quelque chose" et dépendent du problème industriel rencontré. Or, le *e-commerce* correspond à un tout nouveau genre d'activité industrielle adapté à l'ère d'Internet, comme le précise l'ouvrage de Hompel et al. [8]. Un défi majeur pour les détaillants en ligne est la difficulté à répondre à l'augmentation des volumes de commandes, tout en satisfaisant aux attentes de service toujours plus exigeantes. Le coût de fonctionnement de ces centres de distribution est considérable. Un centre de distribution spécialisé dans le commerce en ligne reçoit généralement un grand nombre de petites commandes passées par des particuliers via téléphone, fax ou internet. Ces commandes portent souvent sur seulement 1 à 3 articles. Par exemple, dans les entrepôts d'Amazon en Allemagne, une commande contient en moyenne seulement 1,6 articles selon Boysen et al. [4]. Elles doivent être traitées et expédiées immédiatement après leur réception. En effet, la livraison le lendemain, voire le jour même, constitue une promesse essentielle de nombreux commerçants en ligne dans le segment *business to consumers (B2C)* puisque la fidélité des clients y est fortement liée. Une telle efficacité n'est pas atteignable avec le modèle d'entrepôts traditionnels dont les déplacements des humains représentent jusqu'à 55 % des coûts d'exploitation totaux d'un entrepôt, selon l'article de De Koster et al. [6].

C'est pour répondre à ces exigences que les RMFS ont été conceptualisés par Jünemann en 1989 et brevetés aux États-Unis par *KIVA Systems Inc.*, société ensuite rachetée par *Amazon* et rebaptisée *AmazonRobotics*. Ce système est composé de robots dotés de mécanismes permettant de soulever et de transporter des étagères mobiles de stockage. La bonne orientation des robots est permise par la structuration de l'entrepôt sous forme de grille, où chaque case est identifiée par un code-barres. Les robots se repèrent grâce à la lecture par une caméra intégrée des codes-barres en continu. Les *pods* sont transportés de la zone de stockage jusqu'aux stations de travail. Au niveau de ces postes, plusieurs commandes sont préparées simultanément par un préparateur qui scanne les items apportés par les robots et les dépose dans des bacs correspondants aux commandes en cours. Une description plus détaillée du système est disponible dans les travaux de Wurman et al. [20] et d'Enright [7]. Apporter les stocks aux préparateurs de commandes plutôt que de les laisser se déplacer jusqu'à eux peut doubler leur productivité selon Wurman et al. [20].

Les RMFS présentent de nombreux avantages comparés aux entrepôts classiques et aux autres types d'automatisations dont une présentation détaillée est disponible au sein des revues [16] et [1] (comme les *AS/RS*, *AVS/RS*, ou encore les systèmes *A-frame*). En effet, les robots peuvent accéder à l'ensemble de l'entrepôt et desservir n'importe quelle station

de prélèvement ou de réapprovisionnement, ce qui réduit la dépendance à la main-d'œuvre et limite les retards. Cette organisation facilite le suivi des responsabilités, diminue les erreurs en rehaussant la continuité des opérations, grâce au grand nombre de robots opérant simultanément. La modularité apportée par cette organisation permet d'ajuster en continu la configuration pour répondre aux variations de la demande, tandis que son extensibilité rend possible l'adaptation de la taille de l'installation à l'éventuelle croissance de l'entreprise.

1.2 Éléments de la problématique

Dans un contexte où la compétitivité des acteurs du *e-commerce* repose sur la rapidité de traitement des commandes, l'efficacité physique des systèmes RMFS (*Robot Mobile Fulfillment Systems*) ne suffit plus. Si la robotisation accélère le mouvement, c'est l'intelligence algorithmique pilotant ces robots qui détermine la productivité réelle de l'entrepôt. La problématique centrale de ce mémoire réside dans la prise de décision opérationnelle : comment orchestrer le flux des commandes et le déplacement des étagères pour minimiser les temps improductifs ? Cette section détaille les défis inhérents à l'allocation des ressources dans un système semi-automatisé, en mettant en lumière la complexité combinatoire, l'interdépendance des décisions et les contraintes du temps réel.

La gestion opérationnelle d'un système RMFS repose sur plusieurs problèmes de décision distincts mais intrinsèquement liés. Isoler ces problèmes revient souvent à sous-optimiser la performance globale du système. Nous nous concentrons ici sur les deux décisions majeures que sont l'allocation des commandes aux stations de travail (*Order Batching* et *Pick Order Assignment*) et l'allocation des étagères aux stations (*Pick Pod Selection*). Le premier niveau de décision concerne le flux entrant des commandes clients. Chaque commande est constituée d'une liste d'articles dispersés dans l'entrepôt. L'objectif n'est pas seulement d'affecter une commande à une station disponible, mais de constituer des lots de commandes intelligents. La problématique réside dans la recherche de synergies entre les commandes. Si deux commandes requièrent le même produit ou des produits situés sur la même étagère, il est optimal de les traiter à la même station simultanément. Cela permet d'amener l'étagère une seule fois pour satisfaire plusieurs demandes, augmentant ainsi le nombre d'items cueillis par étagère. Cependant, cette agrégation est contrainte par la capacité des stations. Une mauvaise allocation entraîne une dispersion des ressources : plusieurs robots devront apporter plusieurs étagères différentes pour des produits qui auraient pu être groupés. Le second niveau de décision concerne le choix des étagères. Dans un entrepôt typique, un même item est stocké sur plusieurs étagères différentes pour éviter les ruptures de stock locales et réduire les temps de

trajet. Lorsqu'une commande nécessite un article X, le système doit choisir quelle étagère contenant X mobiliser. Ce choix n'est pas trivial. Il ne s'agit pas simplement de sélectionner l'étagère la plus proche. Il faut privilégier l'étagère qui contient l'article X, mais aussi potentiellement les articles Y et Z nécessaires à d'autres commandes en cours de traitement à la même station. La littérature scientifique et les systèmes industriels classiques ont longtemps traité ces problèmes de manière séquentielle pour réduire la complexité de calcul. Dans une approche séquentielle, le système décide d'abord quelles commandes affecter à une station, puis, une fois cette décision figée, cherche les meilleures étagères pour satisfaire ces commandes. Cette méthode peut forcer le système à déplacer des étagères sous-optimales car le regroupement des commandes initial n'a pas pris en compte la position réelle des stocks. La problématique que nous soulevons dans ce mémoire est celle de la pertinence d'une approche intégrée. En utilisant la programmation mathématique pour résoudre conjointement l'allocation des commandes et des étagères, nous cherchons à atteindre un optimum global.

L'optimisation mathématique doit s'opérer sous des contraintes physiques strictes qui définissent la nature semi-automatisée du système. Le nombre de robots est fini et chaque déplacement d'étagère provoque l'indisponibilité d'un robot. Si l'algorithme demande trop de déplacements pour peu d'articles, alors le système sature, créant une file d'attente d'étagères et immobilisant les préparateurs humains. Les stations de travail sont les points de convergence. Elles ont une capacité limitée et l'allocation ne doit donc pas engorger une station tout en laissant une autre inactive.

La dernière dimension de notre problématique est temporelle. Un entrepôt pour le *e-commerce* est un environnement hautement dynamique et stochastique. Les commandes arrivent en continu et de manière imprévisible à l'entrepôt. La disponibilité des robots et la position des étagères changent seconde après seconde. Un modèle d'optimisation statique, qui calculerait une solution parfaite pour la journée entière, serait obsolète dès la première heure. En résumé, la problématique de ce mémoire consiste à déterminer comment la programmation mathématique en nombres entiers peut capturer la complexité couplée de l'allocation des commandes et des étagères, afin de maximiser le débit de l'entrepôt. La comparaison des formulations "flexible" et "rigide" servira à isoler la structure qui favorise ou pénalise cette performance.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de proposer une approche d'optimisation visant à maximiser le débit de commandes dans un entrepôt de type RMFS, caractérisé par son

dynamisme et ses incertitudes. Pour ce faire, nous nous concentrons sur la résolution conjointe et en temps réel de deux problèmes opérationnels : l'allocation des commandes aux stations (*Pick Order Assignment*) et la sélection des étagères de stockage (*Pick Pod Selection*).

Pour répondre à cette problématique, ce travail de recherche s'articule autour de trois objectifs spécifiques :

1. Développer un modèle de décision simultanée : Il s'agit de formuler un programme mathématique en nombres entiers capable de résoudre conjointement l'allocation des commandes et des étagères. L'enjeu est de créer une formulation "flexible" qui capture les synergies entre les ressources sans figer prématurément les associations.

2. Quantifier les gains de l'approche intégrée : Nous comparerons l'efficacité de notre modèle simultané face à une approche séquentielle classique. L'objectif est de démontrer que l'optimisation globale permet une meilleure utilisation des ressources (robots et stations) qu'une prise de décision séquentielle.

3. Évaluer la robustesse de deux formulations : Nous confronterons le modèle flexible développé dans ce mémoire à un modèle simultané plus rigide, existant dans la littérature. Cette comparaison, effectuée au sein d'une simulation réaliste, vise à déterminer quelle structure de modélisation s'adapte le mieux aux aléas et à l'arrivée continue de commandes dans l'entrepôt.

En somme, ces objectifs visent à identifier la stratégie d'allocation la plus performante pour piloter les systèmes RMFS en temps réel, fournissant ainsi une base solide pour de futures recherches algorithmiques dans le domaine de la logistique.

1.4 Plan du mémoire

Ce travail de recherche est structuré en trois chapitres principaux, reflétant le cheminement allant de la compréhension théorique du problème à sa résolution pratique et son évaluation.

La première partie dresse un état de l'art du domaine. Après avoir contextualisé les défis logistiques imposés par l'essor du commerce en ligne, nous nous concentrons sur la revue de la littérature existante concernant l'optimisation des systèmes RMFS. Ce chapitre permet d'identifier les limites des approches actuelles et de justifier la pertinence de l'utilisation de la programmation mathématique pour la gestion des entrepôts semi-automatisés.

Le deuxième chapitre expose notre contribution principale : la formalisation mathématique

des problèmes de décision. Nous y présentons les différents modèles d'optimisation développés pour l'allocation conjointe des commandes et des étagères. Nous décrirons spécifiquement la formulation "flexible" que nous proposons et le modèle "rigide" issu de la littérature.

Le troisième chapitre est consacré à la présentation du cadre expérimental et à l'analyse des résultats expérimentaux. Nous y détaillons le fonctionnement du framework de simulation utilisé pour reproduire la complexité d'un entrepôt réel. Nous y confrontons aussi les différents modèles au sein de cette simulation selon plusieurs scénarios. Cette analyse comparative nous permettra d'évaluer l'efficacité des approches simultanées face aux approches séquentielles et de discuter de la robustesse du modèle flexible dans un contexte dynamique et incertain.

CHAPITRE 2 REVUE DE LITTÉRATURE

L'optimisation des systèmes RMFS (*Robotic Mobile Fulfillment Systems*) a fait l'objet d'une littérature croissante ces dernières années, motivée par la nécessité d'accroître la réactivité logistique face à l'essor du commerce en ligne. Ce chapitre se propose de synthétiser les travaux existants afin de situer précisément notre contribution dans le paysage scientifique actuel.

Nous débuterons cette analyse par une classification des différentes décisions opérationnelles qui régissent ces entrepôts semi-automatisés. Nous justifierons ensuite notre focalisation spécifique sur les problèmes d'allocation des commandes et des étagères, en nous appuyant sur des études d'impact quantitatives. Enfin, nous examinerons l'évolution des méthodes de résolution, en distinguant les approches séquentielles des approches conjointes, ainsi que les contextes statiques des contextes dynamiques. Cet état de l'art nous permettra d'identifier les limites des modèles actuels et de définir le positionnement de notre recherche.

2.1 Décisions liées à la préparation des commandes dans le cas particulier des RMFS

Dans le contexte du commerce en ligne, les entrepôts semi-automatisés de type RMFS représentent un enjeu d'optimisation majeur. Selon Enright et al. [7], la performance de ces systèmes repose sur la satisfaction d'une double fonction objective : sur le plan opérationnel, il s'agit de maximiser le taux d'occupation des opérateurs humains ; sur le plan stratégique, l'objectif est de minimiser les investissements en équipements, notamment la taille de la flotte de robots.

Pour atteindre ces objectifs, le système doit orchestrer en temps réel une multitude de décisions d'allocation et de planification : gestion des itinéraires, équilibrage de charge aux stations et synchronisation des flux de *pods*. La complexité et l'interconnexion de ces décisions rendent une optimisation globale et instantanée impossible. C'est pourquoi la littérature, notamment Wurman et al. [20] et Merschformann et al. [12], propose de décomposer ce processus en plusieurs sous-problèmes distincts (voir FIGURE 2.1) :

1. ***Pick Order Assignment (POA)*** **:** À l'arrivée des commandes dans le *backlog*, celles-ci doivent être constituées en lots et assignées aux stations de travail disponibles.
2. ***Task Allocation (TA)*** **et** ***Pod Selection (PPS)*** **:** Une fois les commandes affectées, le système doit identifier les *pods* contenant les articles requis et les allouer aux

robots disponibles. Cette étape couple la sélection de l'étagère (PPS) et l'affectation du robot (TA).

3. ***Path Planning (PP)*** **:** Les robots doivent ensuite naviguer à travers l'entrepôt pour acheminer les *pods*, nécessitant un calcul de trajectoire minimisant la distance et évitant les collisions.
4. ***Pod Repositioning (PR)*** **:** Une fois le prélèvement effectué, l'étagère ou *pod* doit être retournée en zone de stockage, potentiellement à un nouvel emplacement optimal.

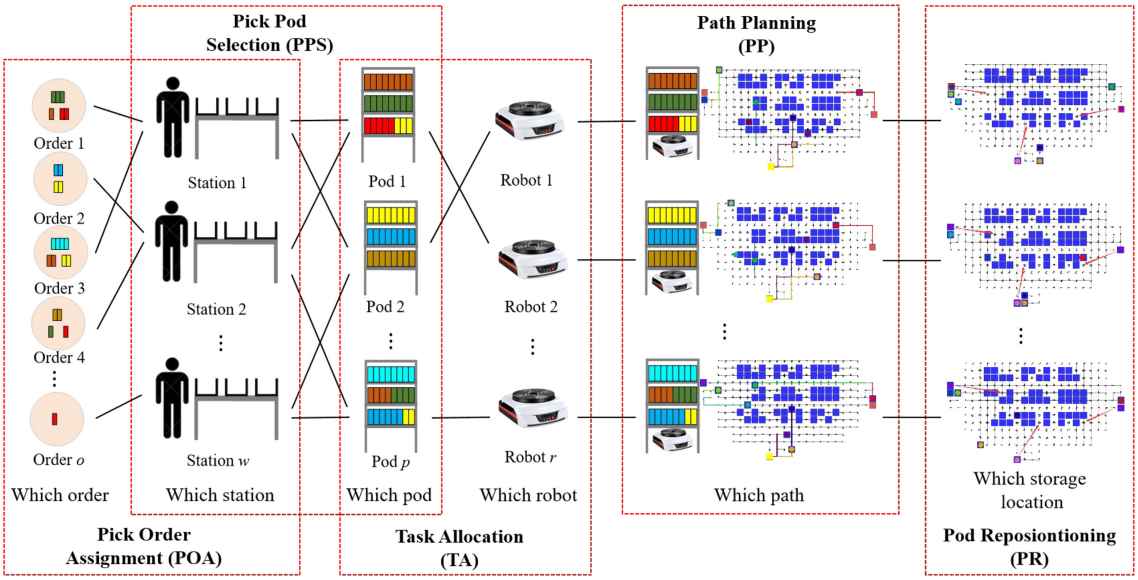

FIGURE 2.1 Schématisation des problèmes de décisions au sein d'un entrepôt RMFS dans le cadre de la préparation des commandes, tirée de l'article de Jiao et al. [10]

2.2 Justification de l'étude des problèmes d'allocation

Le choix de se concentrer spécifiquement sur les problèmes d'allocation des commandes (POA) et de sélection des étagères (PPS) est motivé par leur impact prépondérant sur la performance globale du système. L'étude menée par Merschformann [12] fournit une validation empirique de cette hiérarchie d'importance. En simulant un environnement RMFS réaliste, l'auteur a comparé l'efficacité de diverses règles de décision pour chacun des sous-problèmes identifiés précédemment.

La méthodologie consistait à tester six combinaisons d'heuristiques et à calculer le rapport de performance entre la meilleure et la moins bonne règle pour chaque décision. Les résultats, présentés dans la TABLE 2.1, sont sans équivoque : les décisions POA et PPS affichent des rapports significativement supérieurs à 1. Cela indique une forte sensibilité du système

à la qualité de ces décisions : changer de stratégie d'allocation pour les commandes ou les *pods* modifie drastiquement le débit de l'entrepôt. À l'inverse, l'optimisation des autres sous-problèmes offre des gains marginaux.

TABLEAU 2.1 Résultats de l'expérience décrite et menée dans l'article de Merschforman et al. [12] présentant les rapports de la meilleure et de la pire règle de décision en terme de throughput rate pour chaque problème de décision

							Mult. $\left(\frac{\text{best}}{\text{worst}}\right)$
POA	Common-Lines 50.93%	Due-Time 41.93%	Fast-Lane 76.13%	FCFS 41.81%	Pod-Match **81.18%**	Random **41.71%**	1.946
ROA	Random **53.71%**	Pod-Batch **57.99%**					1.080
PPS	Age 61.50%	Demand 52.70%	Lateness **48.63%**	Nearest **62.16%**	Pile-on 59.82%	Random 48.88%	1.278
RPS	Class 56.16%	Nearest 58.42%	Emptiest **59.63%**	Least-Demand 57.71%	Random **47.56%**		1.254
PSA	Class 55.91%	Fixed 54.08%	Nearest **58.79%**	Random **53.60%**	Station-Based 55.70%		1.097

Une analyse plus fine de la variabilité des résultats (voir FIGURE 2.2) confirme cette tendance : la dispersion des performances est bien plus marquée pour POA et PPS, soulignant qu'il s'agit là des véritables leviers d'optimisation du RMFS.

2.3 État de l'art sur l'optimisation conjointe POA et PPS

Traditionnellement, l'allocation des *pods* et des commandes est traitée de manière séquentielle dans la littérature. Cependant, cette approche découplée ignore les synergies potentielles. Optimiser l'allocation des commandes en fonction de la disponibilité des étagères (et inversement) peut offrir un avantage stratégique majeur.

2.3.1 Approches statiques (*Offline*)

Les premiers travaux sur l'optimisation conjointe se sont placés dans un contexte hors ligne, où toutes les commandes sont connues à l'avance. Boysen et al. [3] ont proposé un modèle linéaire pionnier visant à séquencer simultanément l'arrivée des étagères et le traitement des commandes pour une station unique. Leur objectif était de minimiser les visites d'étagères pour assurer la continuité du travail. Ils ont démontré qu'une telle optimisation pouvait réduire la flotte de robots nécessaire de moitié par rapport à des règles simples. Toutefois,

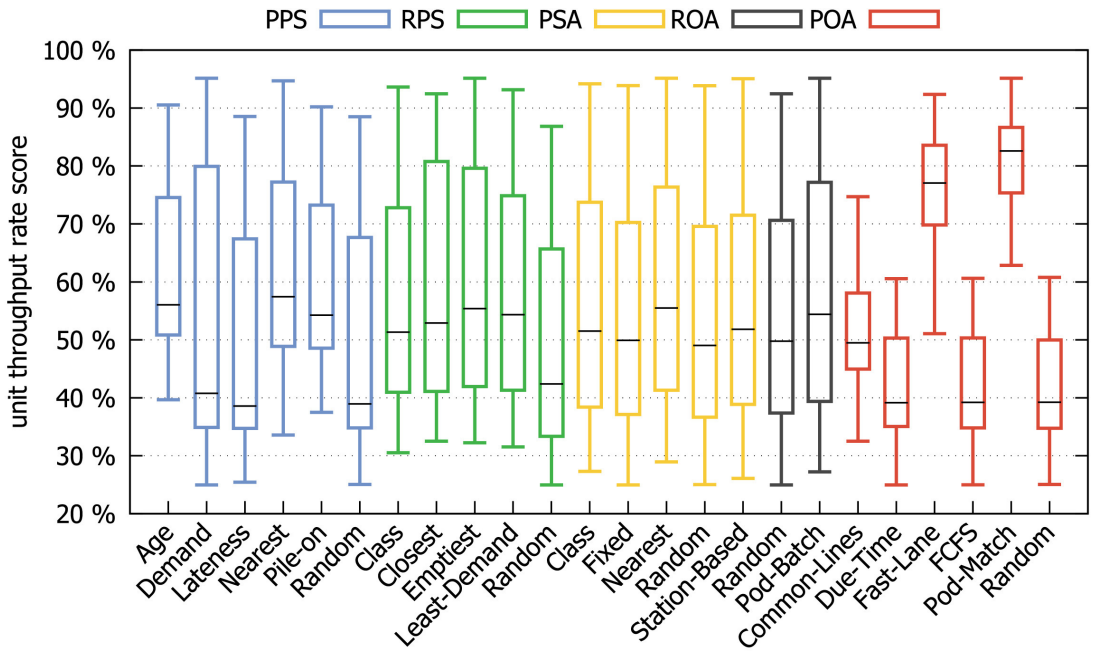

FIGURE 2.2 Résultats sous forme de boîtes à moustaches de l'expérience décrite et menée dans l'article de Merschforman et al. [12] pour chaque problème de décision et ses règle de décision associées

leur modèle simplifie la réalité en ne considérant qu'une seule station et en ignorant les contraintes de disponibilité des stocks.

Yang et al. [23] ont étendu cette approche au cas multi-stations, mais en négligeant les conflits d'accès aux étagères partagées. Par la suite, Valle et al. [18] et Wang et al. [19] ont affiné ces modèles en intégrant des contraintes de capacité (nombre d'articles par *pod* limité) et des tournées de *pods* desservant plusieurs stations. Bien que performants, ces modèles restent limités par leur nature statique, inadaptée à la réalité opérationnelle d'un entrepôt recevant des commandes en continu.

2.3.2 Approches dynamiques (*Online*)

Comme le soulignent Wurman et al. [20], les RMFS sont des systèmes intrinsèquement dynamiques. L'absence de fenêtre temporelle pour le calcul hors ligne impose des prises de décision en temps réel.

Xie et al. [22] ont été les premiers à introduire une prise de décision en ligne pour des stations multiples, intégrant la possibilité de diviser une commande entre plusieurs stations. Plus récemment, Jiao et al. [10] ont proposé un modèle traitant le même problème, mais

avec des hypothèses plus réalistes : commandes multi-items et capacité limitée des étagères. Contrairement aux approches séquentielles, leur objectif maximise le nombre de commandes traitées tout en minimisant l'usage des étagères. La résolution est déclenchée par un seuil de capacité aux stations. Cependant, ce modèle présente une certaine rigidité : le grand nombre de variables reliant items, commandes et étagères le rend difficilement mis à l'échelle, limitant les expériences à des instances de petite taille. De plus, sa structure fige les associations précocement, ce qui peut s'avérer sous-optimal en cas d'imprévus.

2.3.3 Positionnement et contribution

La littérature récente a mis en évidence les bénéfices théoriques de l'optimisation conjointe des problèmes d'affectation des commandes et des étagères. Cependant, la grande majorité de ces études se limite à des environnements statiques (*offline*), où l'ensemble des données est connu a priori. Les travaux traitant cette problématique en contexte dynamique (*online*) restent rares. Si une approche existante (Jiao et al. [10]) propose bien un modèle de résolution simultanée en temps réel, celle-ci repose sur une formulation structurellement « rigide » avec un grand nombre de variables, figeant les associations entre les ressources de manière précoce.

Le manque identifié dans la littérature réside donc dans le besoin d'une formulation plus flexible, capable de s'adapter aux aléas et aux indisponibilités récurrentes des robots. Ce mémoire vise ainsi à repousser cette limite en proposant un nouveau modèle mathématique flexible et en le confrontant à l'approche rigide existante au sein d'une simulation réaliste, afin d'identifier la stratégie la plus résiliente pour les systèmes RMFS.

CHAPITRE 3 MODÈLES MATHÉMATIQUES POUR L'ALLOCATION SIMULTANÉE DE COMMANDES ET ÉTAGÈRES VERS LES STATIONS DE TRAVAIL

Afin d'objectiver la performance des stratégies d'allocation dans notre environnement de simulation, il est nécessaire de définir au préalable le cadre d'évaluation ainsi que les formulations exactes des solutions testées. Cette section présente dans un premier temps les métriques clés retenues, notamment le débit de commandes (*throughput*), qui serviront de baromètre pour mesurer l'efficacité opérationnelle de l'entrepôt. Dans un second temps, nous détaillerons la formalisation mathématique des deux modèles d'optimisation au cœur de notre étude comparative : le modèle que nous qualifions de « flexible » que l'on appelle **MIP**, développé pour cette recherche, et le modèle « rigide » que l'on appelle **MIPJ**, issu de la littérature. Cette mise en parallèle de leurs structures et de leurs contraintes respectives permettra de mieux appréhender leurs comportements face aux dynamiques de l'entrepôt.

3.1 Métriques considérées pour calculer les performances de l'entrepôt

Nous distinguons principalement les indicateurs suivants :

Throughput Rate Il peut s'exprimer en *Unit throughput rate* (nombre d'unités prélevées par heure) ou en *Pick order throughput rate* (nombre de commandes complétées par heure). Plus cette valeur est élevée, plus le système est efficace.

Throughput Time Le temps nécessaire pour traiter une commande ou une unité .

Order Turnover Time Le délai moyen s'écoulant entre l'arrivée d'une commande dans le *backlog* et sa finalisation.

Distance totale parcourue La somme des distances effectuées par l'ensemble de la flotte de robots.

Tardiness La fraction des commandes en retard ou le délai moyen entre l'échéance promise et l'exécution réelle.

Pile-on Le nombre moyen d'unités prélevées sur une même étagère à chaque visite à une station.

Temps d'inactivité Le temps durant lequel les stations de préparation sont à l'arrêt faute d'étagères disponibles.

Bien que le *throughput rate* soit la métrique résumant le mieux l'efficacité globale de l'entrepôt, il s'agit d'une résultante complexe du système, difficile à intégrer directement comme fonction objectif dans un modèle linéaire. C'est pourquoi il est nécessaire de s'appuyer sur des métriques intermédiaires plus faciles à modéliser.

Or, comme le démontrent les expériences de Merschformann et al. [12], ces différents indicateurs sont fortement corrélés entre eux. La matrice de corrélation présentée dans le TABLEAU 3.1 illustre ces interdépendances.

TABLEAU 3.1 Corrélation des différents indicateurs de performances mesurés dans le cadre d'expériences menées dans l'article de Merschformann et al. [12]

	Unit throughput	Order throughput	Order turnover time	Distance traveled	Order offset	Late orders	Pile-on	Station idle time	μ	σ
Unit throughput	-	-	-	-	-	-	-	-	0.556	0.189
Order throughput	1.000	-	-	-	-	-	-	-	241.963	82.234
Order turnover time	-0.950	-0.950	-	-	-	-	-	-	3549.625	1220.445
Distance traveled	-0.952	-0.952	0.880	-	-	-	-	-	122598.76	19433.86
Late orders	-0.590	-0.591	0.685	0.549	0.684	-	-	-	0.187	0.115
Pile-on	0.899	0.899	-0.802	-0.796	-0.802	-0.448	-	-	2.438	1.450
Station idle time	-1.000	-1.000	0.950	0.952	0.950	0.591	-0.899	-	0.450	0.186

L'analyse de ces corrélations révèle que les deux leviers principaux influençant le débit sont le *pile-on* et la distance parcourue par les robots, puisque ce sont deux métriques qui sont très corrélées à celle du *throughput*. La distance parcourue par les robots et le *unit throughput* présentent une corrélation de 0.952 en valeur absolue. Celle du *pile-on* et du *unit throughput* est de 0.899. En effet, pour maximiser le nombre de commandes traitées par heure, il est crucial d'assurer la continuité du travail aux stations. Cela passe par une haute disponibilité des robots, obtenue soit en réduisant leurs temps de trajet, soit en augmentant la productivité de chaque trajet (maximisation du *pile-on*). L'impact combiné de ces deux facteurs sur le débit est visualisé sur le nuage de points de la FIGURE 3.1.

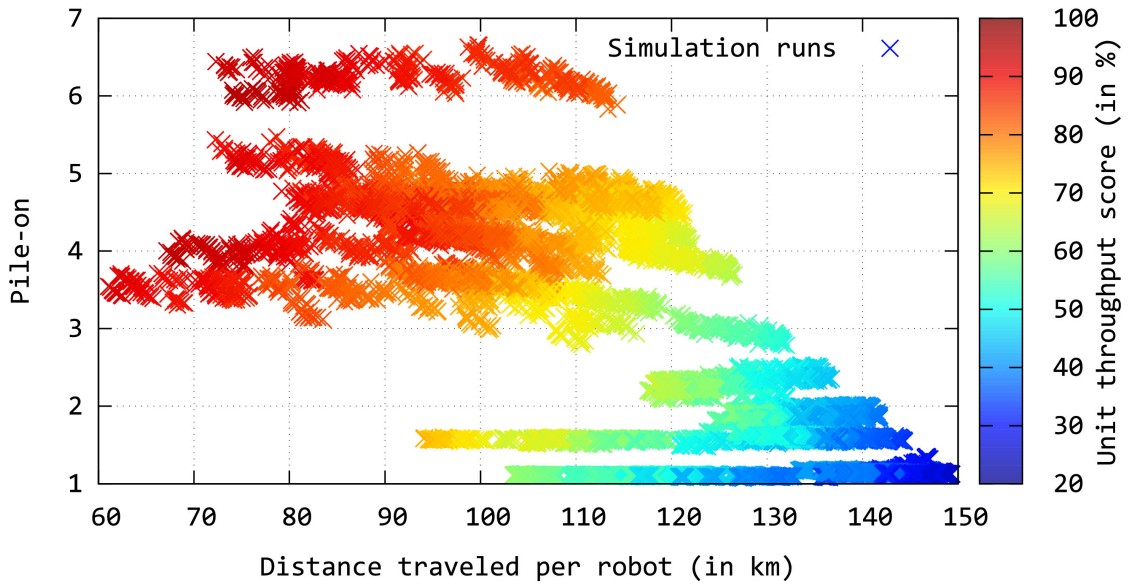

FIGURE 3.1 Nuage de points pour la distance parcourue par robot par rapport au pile-on, coloré par le throughput rate atteint, dans le cadre d'expériences menées dans l'article de Merschformann et al. [12]

Il apparaît que l'optimisation conjointe de ces facteurs peut se traduire mathématiquement par la minimisation du nombre de visites des *pods* aux stations. En effet, minimiser les visites pour un même volume de commandes revient mécaniquement à augmenter le nombre d'articles prélevés par visite (le *pile-on*).

C'est cette logique qui a guidé la conception des modèles présentés dans la section suivante. Notre fonction objectif visera donc à minimiser le nombre de voyages des *pods* (et implicitement maximiser le *pile-on*). Concernant la minimisation pure de la distance parcourue, celle-ci n'a pas été retenue comme objectif prioritaire dans notre modélisation, partant du postulat que les acteurs de l'e-commerce tendent à compenser les coûts de transport interne par l'investissement dans des flottes de robots plus conséquentes.

3.2 Affectation des commandes et des étagères par le modèle MIP

Dans cette section, nous présentons un modèle MIP conçu pour optimiser l'affectation des *pods* et des commandes grâce à une approche par lots. La motivation de ce type de modèle est de pouvoir assigner des commandes de prélèvement aux stations de travail et de baser le renouvellement de ce processus, non plus sur l'arrivée unitaire des commandes à l'entrepôt, comme c'est originellement le cas dans la simulation RawSim-O, mais en testant un déclencheur différent basé sur l'exécution des commandes aux stations. Cela pourrait permettre

d'assurer une meilleure continuité dans le travail effectué aux stations et donc une meilleure performance de l'entrepôt.

Les paramètres et variables sont décrits dans le TABLEAU 3.2. Les variables de décision indiquent si des commandes et des *pods* spécifiques sont affectés à des stations et comment les articles sont prélevés dans les *pods*. Pour chaque commande o et station s, nous introduisons une variable binaire y_{os} indiquant si la commande o est assignée à la station s ($y_{os} = 1$) ou non. De même, pour chaque *pod* p et station s, nous définissons une variable binaire x_{ps}, qui vaut 1 si le *pod* est assigné à la station s, et 0 sinon.

TABLEAU 3.2 Définition des paramètres pour le modèle linéaire basé sur les lots intégrant les décisions POA et PPS

Indices	Définition
i	indice des SKU
o	indice des commandes
p	indice des *pods*
s	indice des stations
Paramètres	Définition
$\mathcal{P}$	Ensemble des *pods*
$\mathcal{S}$	Ensemble des stations de travail
$\mathcal{I}$	L'ensemble des SKUs
$\mathcal{O}$	Ensemble des commandes dans le *backlog*
$\bar{\mathcal{O}}_s$	Ensemble des commandes déjà assignées à la station s
$\bar{\mathcal{P}}_s$	Ensemble de *pods* déjà en cours de prélèvement à la station $s \in S$ ou en chemin vers la station
n_{ip}	Nombre d'articles i dans le *pod* p
n_{io}	Nombre d'articles i demandés par la commande o
C_s	Capacité de la station s
Variables	Définition
x_{ps}	1 si le *pod* p est assigné à la station s, 0 sinon
y_{os}	1 si la commande o est assigné à la station s, 0 sinon

$$\min \sum_{(p,s)\in\mathcal{P}\times\mathcal{S}} x_{ps} - \sum_{(o,s)\in\mathcal{O}\times\mathcal{S}} y_{os} \tag{3.1}$$

$$\sum_{s\in S} y_{os} \leq 1, \forall o \in \mathcal{O} \tag{3.2}$$

$$\sum_{s\in S} x_{ps} \leq 1, \forall p \in \mathcal{P} \tag{3.3}$$

$$\sum_{p\in\mathcal{P}} n_{ip} \cdot x_{ps} \geq \sum_{o\in\mathcal{O}} n_{io} \cdot y_{os}, \forall (i,s) \in \mathcal{I} \times \mathcal{S} \tag{3.4}$$

$$\sum_{o\in\mathcal{O}} y_{os} \leq C_s, \forall s \in \mathcal{S} \tag{3.5}$$

$$y_{os} = 1, \forall (o,s) \in \bar{\mathcal{O}}_s \times \mathcal{S} \tag{3.6}$$

$$x_{ps} = 1, \forall (p,s) \in \bar{\mathcal{P}}_s \times \mathcal{S} \tag{3.7}$$

$$x_{ps} \in \{0,1\}, \forall p,s \in \mathcal{P} \times \mathcal{S} \tag{3.8}$$

$$y_{os} \in \{0,1\}, \forall o,s \in \mathcal{O} \times \mathcal{S} \tag{3.9}$$

L'objectif (3.1) minimise le nombre total de *pods* assignés aux stations. Les contraintes (3.2) garantissent que chaque commande est assignée à une seule station de préparation de commandes au maximum. Les contraintes (3.3) garantissent que chaque *pod* est assigné à une seule station de travail au maximum. Les contraintes (3.4) garantissent que le nombre d'articles contenus dans les *pods* assignés à une station est suffisant pour traiter les commandes de cette station. Les contraintes (3.5) garantissent que chaque station reçoit au maximum C_s commandes.

Les contraintes (3.6) et (3.7) garantissent que chaque station reçoit toutes les commandes et tous les *pods* attribués (s'ils ont été déjà déplacés par un robot) de manière fixe lors de l'étape de planification précédente dans la planification actuelle. Les contraintes (3.8) et (3.9) garantissent que les variables respectives ne doivent contenir que des valeurs binaires.

Le modèle permet une optimisation dynamique grâce à sa mise à jour régulière en fonction de l'état actuel du système.

Les inconvénients du modèle résident dans le problème de séquencement des commandes et des *pods* dans chaque station, car le nombre de commandes attribuées à chaque station peut être supérieur au nombre de *pods*. Ce problème peut être résolu en résolvant |S| d'autres problèmes d'optimisation à chaque station afin de trouver les séquences optimales. De plus, affecter l'intégralité du *backlog* aux stations lorsque celui-ci est très important ne serait pas plus avantageux car les lots seraient également importants et donc moins similaires que pour

un *backlog* plus petit.

Le modèle présente des difficultés pour déterminer le moment opportun pour déclencher une réoptimisation. Cette décision nécessite un réglage minutieux afin d'équilibrer la réactivité du modèle et sa charge de calcul.

3.3 Allocation des commandes et des *pods* par le modèle MIPJ tiré de l'article de Jiao et al. [10]

Dans cette section, nous présentons le modèle MIPJ conçu par Jiao et al. [10] pour optimiser l'affectation des *pods* et des commandes. La motivation derrière ce modèle est de le comparer avec le modèle précédent.

Les paramètres et variables sont décrits dans le TABLEAU 3.3. Les variables de décision indiquent si des commandes et des *pods* spécifiques sont affectés à des stations et comment les articles sont prélevés dans les *pods*. Pour chaque commande o et station s, une variable binaire y_{os} indiquant si la commande o est traitée à la station s ($y_{os} = 1$) ou non est introduite. De même, pour chaque *pod* p et station s, est définie une variable binaire x_{ps}, qui vaut 1 si le *pod* est requis/déjà présent à la station s, et 0 sinon. Enfin, une variable z_{iops} est introduite et vaut 1 si l'item i de la commande o à la station s est servi par le *pod* p.

Dans ce modèle aussi, l'état du système est mis à jour en temps réel par la simulation, éliminant ainsi le recours à un indice de temps discret t. Au lieu de modéliser explicitement les périodes, l'état actuel du système, comme le nombre de *pods* à chaque station ($\bar{\mathcal{P}}s$), les commandes déjà présentes aux stations ($\bar{\mathcal{O}}s$), les niveaux de stock des *pods* n_{ip} et le nombre d'unités de gestion des stocks (SKU) encore nécessaires pour les commandes n_{io}, est directement extrait de la simulation à tout moment. Cela permet au modèle de prendre des décisions basées sur les informations les plus récentes, garantissant une approche dynamique et réactive pour optimiser l'affectation des *pods* et des commandes, tout en évitant un grand nombre de variables supplémentaires nécessaires à la représentation de plusieurs périodes.

Tableau 3.3 Définition du modèle MIP intégrant les règles POA et PPS

Indices	Définition
i	indice du SKU
o	indice de la commande
p	indice du *pod*
s	indice de la station
Paramètres	**Définition**
$\mathcal{P}$	Ensemble des *pods*
$\bar{\mathcal{P}}_s$	Ensemble des *pods* déjà à la station ou en chemin vers elle $s \in S$
$\mathcal{S}$	Ensemble des stations
$\mathcal{I}$	Ensemble des SKUs
$\mathcal{I}_o$	Ensemble des SKUs de la commande o
$\mathcal{O}$	Ensemble des commandes du *backlog*
C_s	Capacité de la station s
n_{ip}	Nombre de SKU i dans le *pod* p
n_{io}	Nombre de SKU i demandés par la commande o
Variables	**Définition**
x_{ps}	1 si le *pod* p est assigné à la station s, 0 sinon
y_{os}	1 si la commande o est assignée à la station s, 0 sinon
y_{ios}	1 si l'item i de la commande o est assignée à la station s, 0 sinon
z_{iops}	Nombre de SKU i prélevés pour la commande o du *pod* p à la station s

L'objectif (3.10) minimise le nombre total de *pods* assignés à chaque instant aux stations.

$$\text{Minimize} \quad \sum_{(p,s)\in\mathcal{P}\times\mathcal{S}} x_{ps} \tag{3.10}$$

$$\text{s.t.} \quad y_{os} = y_{ios}, \qquad \forall(i,o,s) \in \mathcal{I}_o \times \mathcal{O} \times \mathcal{S} \tag{3.11}$$

$$\sum_{s\in S} y_{os} \leq 1, \qquad \forall o \in \mathcal{O} \tag{3.12}$$

$$\sum_{o\in O} z_{iops} \leq n_{ip} \cdot x_{ps}, \forall(i,p,s) \in \mathcal{I} \times \mathcal{P} \times \mathcal{S} \tag{3.13}$$

$$\sum_{p\in P} z_{iops} = n_{io} \cdot y_{ios}, \forall(i,o,s) \in \mathcal{I} \times \mathcal{O} \times \mathcal{S} \tag{3.14}$$

$$\sum_{o\in O} y_{os} = C_s, \qquad \forall s \in \mathcal{S} \tag{3.15}$$

$$x_{ps} = 1, \qquad \forall(p,s) \in \bar{\mathcal{P}}_s \times \mathcal{S} \tag{3.16}$$

$$\sum_{s\in S} x_{ps} \leq 1, \qquad \forall p \in \mathcal{P} \tag{3.17}$$

$$x_{ps} \in \{0,1\}, \qquad \forall p \in \mathcal{P} \tag{3.18}$$

$$y_{os} \in \{0,1\}, \qquad \forall s \in \mathcal{S} \tag{3.19}$$

$$y_{ios} \in \{0,1\}, \qquad \forall i \in \mathcal{I}_o, o \in \mathcal{O}, s \in \mathcal{S} \tag{3.20}$$

$$z_{iops} \in \{0,1\}, \qquad \forall i \in \mathcal{I}_o, o \in \mathcal{O}, p \in \mathcal{P}, s \in \mathcal{S} \tag{3.21}$$

Les contraintes (3.11) garantissent que si une commande est affectée à une station, alors toutes les lignes de commande doivent également être affectées à cette même station. Si la commande ne peut être affectée à aucune station, c'est-à-dire qu'elle reste en attente (dans le *backlog*), alors aucune de ses lignes ne peut être affectée à une station. Les contraintes (3.12) garantissent que chaque commande peut être affectée à au plus une station. Les contraintes (3.13) sont des contraintes d'inventaire des *pods* qui garantissent que le nombre d'unités de gestion des stocks (SKU) fournies par le pod ne dépasse pas le nombre de SKU stockés dans ce *pod*. Les contraintes (3.14) garantissent que le nombre de SKU fournis par l'ensemble des *pods* pour la station s, contenant le SKU doit être égal au nombre de SKU requis dans les commandes étant affectées à la station s. Les contraintes (3.15) garantissent que le nombre de commandes affectées à une station est égal à la capacité disponible de cette station. Les contraintes (3.16) garantissent que tous les *pods* de l'ensemble $\bar{\mathcal{P}}_s$ sont affectés à la station. L'ensemble $\bar{\mathcal{P}}_s$ comprend les *pods* en cours de prélèvement à la station, ceux en attente dans la zone de file d'attente, ainsi que ceux en transit depuis la zone de stockage vers la file d'attente. Les contraintes (3.17) garantissent que chaque *pod* ne peut être affecté qu'à une seule station. Les contraintes (3.18), (3.19), (3.20), (3.21) définissent le domaine des variables

de décision.

Les grandes différences entre ces deux modèles sont le nombre plus élevé de variables pour MIPJ et la contrainte 3.16 du modèle MIPJ qui reprend les solutions antérieures concernant l'allocation des différentes étagères qui avaient été sélectionnées lors de résolutions précédentes.

CHAPITRE 4 ÉTUDE EXPÉRIMENTALE

Après avoir détaillé la formulation des modèles mathématiques d'allocation, ce chapitre est consacré à leur évaluation empirique. L'objectif est de confronter nos approches d'optimisation (flexible et rigide) à la réalité dynamique et incertaine d'un entrepôt opérationnel. Pour garantir la rigueur de cette validation, nous avons structuré notre démarche expérimentale en trois phases distinctes.

Dans un premier temps, nous présenterons le cadre de simulation utilisé pour nos expériences : le framework open-source RawSimO. Nous décrirons comment cet outil nous permet de reproduire fidèlement les contraintes physiques et temporelles d'un système RMFS, ainsi que les adaptations techniques nécessaires à l'intégration de nos solveurs mathématiques.

Dans un second temps, nous procéderons à une phase de calibrage de la référence (baseline). Avant de tester nos modèles complexes, il est essentiel d'établir un point de comparaison performant. Nous analyserons donc différentes combinaisons d'heuristiques simples afin d'identifier la configuration la plus efficace. Celle-ci servira de témoin pour mesurer le gain réel apporté par l'optimisation mathématique.

Enfin, la dernière partie de ce chapitre exposera le cœur de nos résultats : l'analyse comparative des modèles mathématiques. Nous y confronterons les performances du modèle flexible, du modèle rigide et de la baseline calibrée précédemment, à travers divers scénarios de charge. Cette analyse nous permettra de conclure sur la pertinence et la robustesse de l'allocation conjointe en temps réel.

4.1 Prises de décisions en temps réel à l'aide de la simulation d'un entrepôt RMFS par le framework RawSim-O

4.1.1 Présentation générale du framework

Le framework *RawSim-O*, développé par Merschformann [14], est une solution de simulation *open source* particulièrement adaptée aux objectifs de notre étude. Le TABLEAU 4.1 présente une vue synthétique des caractéristiques de cet outil. Notre travail se concentre sur l'application `RawSim-O.Core`, et plus spécifiquement sur le module `RawSim-O.Core.Control`. Ce dernier contient les classes nécessaires à l'implémentation des différentes règles de gestion régissant un RMFS (voir section 2.1).

Cette simulation repose sur une architecture multi-agents pilotée par des événements discrets.

Tableau 4.1 Résumé du framework RawSim-O

Aspect	**Détails**
Type de simulation	Événements discrets pour systèmes de robots mobiles
Objectif	Étudier les décisions logistiques dans les entrepôts robotisés
Échelle	Peut simuler de petits à grands entrepôts (échelle modifiable)
Extensibilité	Permet d'ajouter de nouvelles stratégies de décision
Langage	Développé en `C#`, avec support pour Visual Studio
Nombre d'applications développées à l'aide de .NET	15
Nombre de dossiers dans l'application RawSim-O.Core	18
Nombre de classes dans RawSim-O.Core.Control	74
Algorithmes développés	Heuristiques
Exécution	Par un terminal ou interface graphique avec visualisation 2D/3D

La FIGURE 4.1 illustre la structure globale de ce fonctionnement événementiel. L'environnement est initialisé via une série de fichiers de configuration, édités par l'utilisateur ou générés procéduralement. Le déroulement de la simulation est orchestré par un noyau central (le simulateur) chargé de la mise à jour fréquente des agents. Ces agents, qui agissent comme contrôleurs de la simulation, collaborent pour exécuter les décisions opérationnelles et transmettre l'état du système au module de visualisation.

Comme résumé dans le TABLEAU 4.1, RawSim-O a été conçu pour analyser la complexité des flux logistiques robotisés : ordonnancement du picking, affectation des pods, routage ou encore gestion des files d'attente. Sa force réside dans sa flexibilité et sa capacité à reproduire fidèlement le comportement cinématique des robots, permettant ainsi d'évaluer l'impact précis des décisions sur la performance globale. Ses fonctionnalités clés incluent une architecture modulaire facilitant l'intégration de nouvelles politiques, une visualisation 2D/3D, et une gestion fine des événements temporels liés aux agents (robots, *pods* et stations).

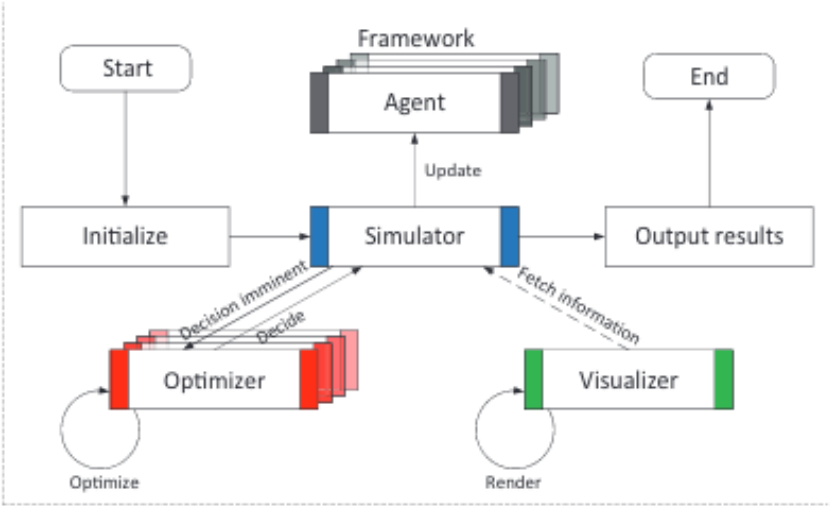

FIGURE 4.1 Vue globale du fonctionnement du framework RawSim-O, tirée de l'article de Merschformann et al. [14]

De plus, au-delà de son intérêt théorique pour la simulation, ce framework possède la capacité d'être interfacé avec des systèmes physiques réels, comme démontré par Xie et al. [21].

4.1.2 Présentation détaillée de RawSim-O

La configuration de l'entrepôt utilisée dans cette étude s'appuie sur les travaux de Lamballais et al. [11]. La FIGURE 4.2 permet de visualiser l'agencement typique : la zone de stockage centrale contient les pods (carrés bleus), autour desquels naviguent les robots (carrés verts). Les stations de travail (picking) sont situées à droite en rouge, tandis que les stations de remplissage (replenishment) se trouvent à l'opposé, en jaune. Cette modélisation est paramétrique, permettant de simuler des instances de tailles variables.

Au sein du simulateur, les décisions s'enchaînent selon une chronologie précise, détaillée en FIGURE 4.3. Le processus débute par la mise à jour de l'agent superviseur ("*Manager*"). L'arrivée de nouvelles commandes dans le *backlog* déclenche la décision d'affectation des commandes (POA). Une heuristique interne alloue alors les commandes, une à une, aux stations de travail disponibles, générant ainsi des listes de tâches d'extraction pour chaque station.

À cette étape, l'association entre les articles demandés et les *pods* spécifiques n'est pas encore figée. Cette liaison relève de la décision de sélection des étagères (PPS). Dans l'architecture RawSim-O, la décision PPS est intrinsèquement liée à l'allocation des tâches (TA). Elle est déclenchée lorsqu'un robot devient disponible : cherchant une tâche à effectuer, le robot se

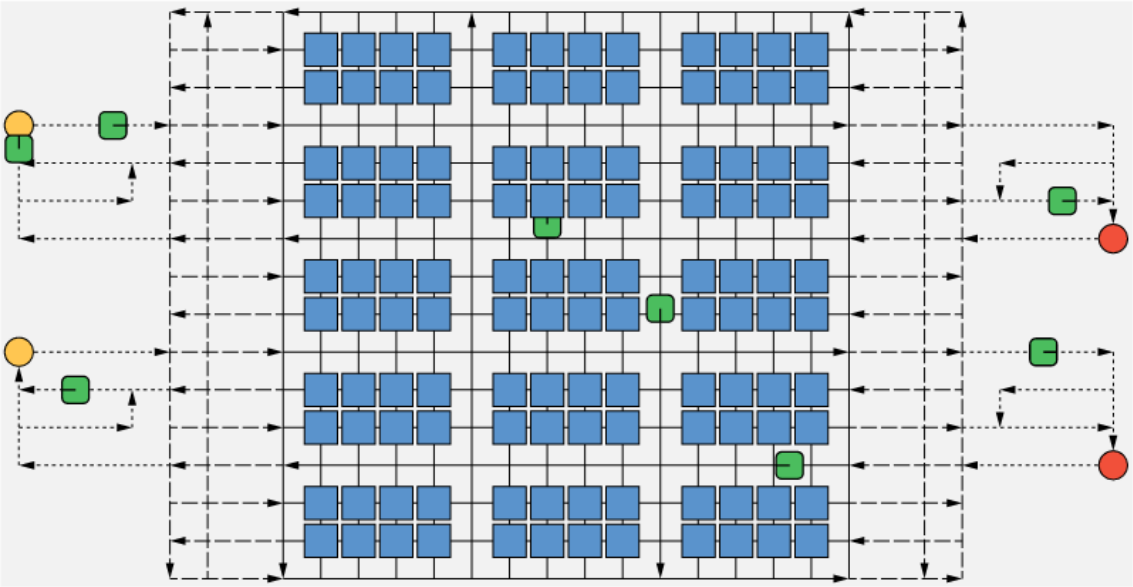

FIGURE 4.2 Représentation du design du RMFS au sein de la simulation RawSim-O tirée de l'article de Merschformann et al. [14]

voit attribuer un *pod* selon une règle de décision prédéfinie. C'est à cet instant précis que s'opère la correspondance entre les besoins des stations et le contenu du *pod* sélectionné.

Enfin, une fois le couple robot-*pod* formé et la destination fixée, le calcul de l'itinéraire est assuré par la décision de cheminement (PP). Pour cette tâche, Merschformann et al. [13] recommandent l'algorithme WHCA*, qui offre le meilleur compromis en évitant les collisions et impasses tout en respectant les contraintes cinématiques, bien que sa mise à l'échelle reste limitée comparée à d'autres approches .

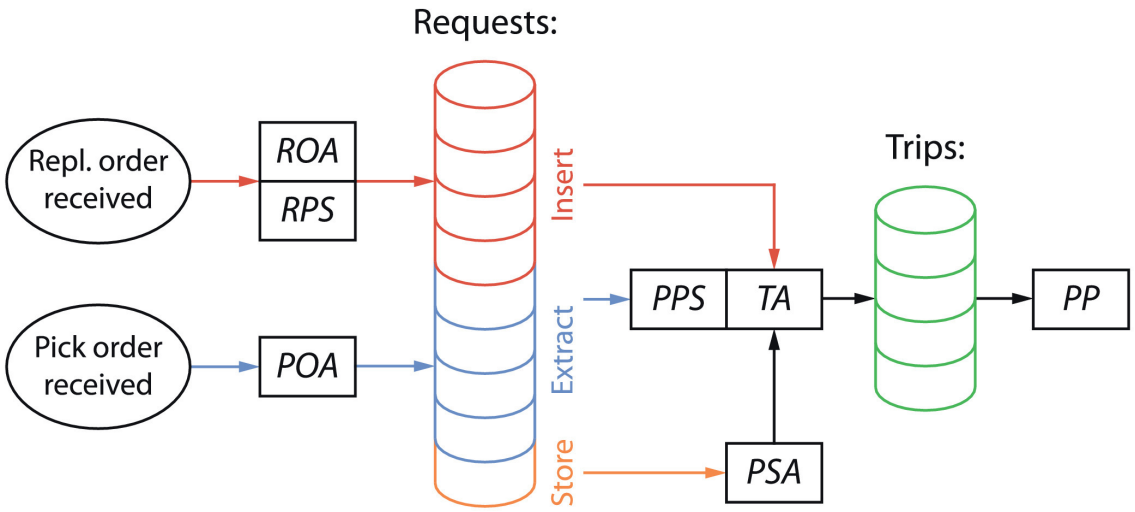

FIGURE 4.3 Schéma de la chronologie des décisions prises dans le RMFS simulé par le framework RawSim-O, tirée de l'article de Merschformann et al. [14]

Il est important de noter que l'entrepôt simulé opère sous une contrainte de charge constante. Le *backlog* est maintenu à un niveau fixe. Dès qu'une décision POA assigne une commande à une station, une nouvelle commande entre immédiatement dans la file d'attente, maintenant le système sous pression continue.

4.2 Paramétrage et calibrage des heuristiques de décision

Cette section détaille les expériences visant à identifier la combinaison d'heuristiques la plus performante pour servir de référence (*baseline*).

Le TABLEAU 4.2 recense les paramètres de configuration de l'entrepôt simulé, dont un visuel est proposé en FIGURE 4.4.

TABLEAU 4.2 Liste des paramètres choisis au sein de la simulation RawSim-O

Paramètre	**Valeur / Description**
Dimensions de l'entrepôt	8 allées horizontales × 6 allées verticales
Nombre de *pods*	224 (Capacité : 500 items/*pod*)
Temps de simulation	48h
Stations de sortie	4 (Capacité : 12 commandes)
Stations d'entrée	4
Nombre de robots	32
Aléatoire	Oui

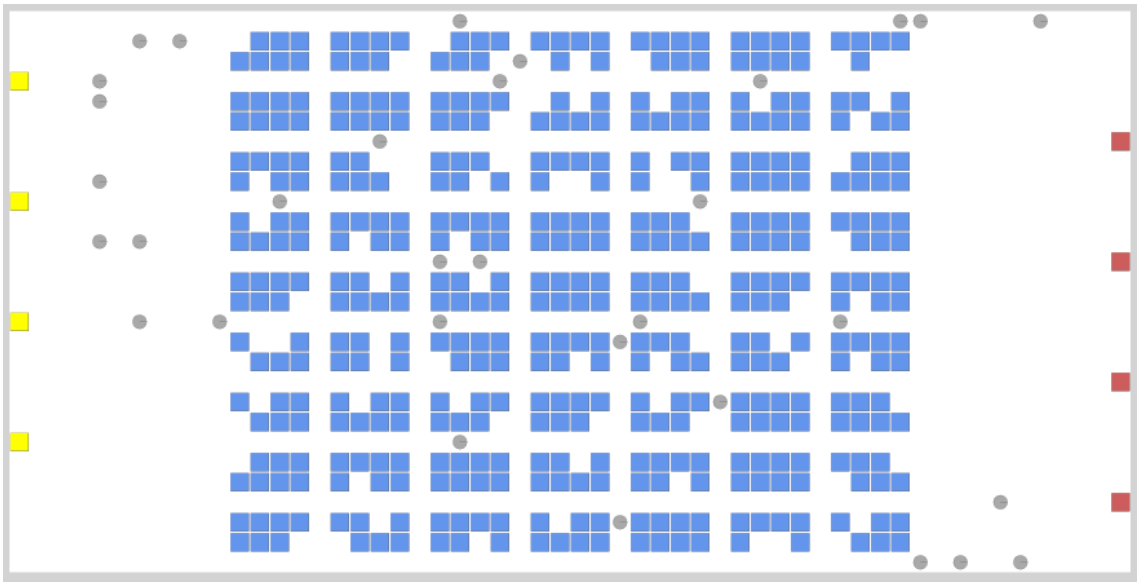

FIGURE 4.4 Capture d'écran de l'entrepôt RawSim-O configuré pour les expériences heuristiques

Pour faciliter la lecture des résultats suivants, voici la définition des acronymes utilisés dans les tableaux de mesures :

- **OP (*Orders Placed*) :** Nombre moyen de commandes placées par heure.
- **OH (*Orders Handled*) :** Nombre moyen de commandes traitées par heure.
- **OTHT (*Order Throughput Time*) :** Temps moyen de traitement d'une commande (en secondes).
- **OTOT (Order Turnover Time) :** Temps moyen de rotation d'une commande dans le système (en secondes).
- **DT (*Distance Travelled*) :** Distance totale parcourue par la flotte de robots (en mètres).
- **IP (*Item Pile-on*) :** Nombre moyen d'articles prélevés par pod et par visite.

4.2.1 Sélection de la meilleure règle d'affectation des commandes (POA)

Dans cette première expérience, nous cherchons la règle POA la plus performante, toutes les autres décisions étant fixées à leurs valeurs par défaut. Les résultats sont exposés dans le TABLEAU 4.3.

TABLEAU 4.3 Comparaison des règles POA (autres règles par défaut)

Règle POA	**OP**	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
Default (DueTime)	10,8	4,0	31976,5	84143,7	46355,0	1,3
FCFS	11,0	4,2	32969,4	86634,9	46365,7	1,2
Foresight	12,5	6,5	20317,7	75103,5	46518,9	1,9
Lines In Common	11,8	5,6	30503,0	74288,1	46403,9	1,8
Near Best Pod (Manh.)	12,1	4,3	33604,4	34326,8	46475,9	1,1
Near Best Pod (Short.)	11,2	4,2	32831,8	33565,0	46483,7	1,2
Pod Matching	**14,1**	**9,7**	**16548,1**	**60654,6**	**46405,5**	**2,2**
Queue Order	10,6	5,3	24937,5	67530,5	46291,2	1,5
Random	11,2	4,5	30666,2	31370,2	46528,1	1,2
Related Order	12,1	3,6	35310,8	36110,2	46362,0	1,3
Workload (High.)	11,3	4,6	30515,6	31204,2	46404,2	1,2
Workload (Low.)	10,7	3,9	36722,7	37515,3	46421,2	1,2

Analyse : La règle *Pod Matching* obtient le meilleur score. Son temps de traitement (OTHT) est environ 0,6 fois celui de la meilleure règle concurrente (*Queue Order*, 24 937 s). Cette performance s'appuie sur un *Pile-on* (IP) élevé de 2,2, indiquant un meilleur groupement des commandes.

4.2.2 Sélection de la meilleure règle de sélection des *pods* (PPS)

Nous fixons désormais POA sur *Pod Matching* et faisons varier la règle PPS. Les résultats sont présentés dans le TABLEAU 4.4.

TABLEAU 4.4 Comparaison des règles PPS (avec POA = Pod Matching)

Règle PPS	**OP**	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
Brute Force	**16,4**	**12,0**	**10140,4**	**49142,7**	**46644,9**	**3,6**
Random	13,3	8,8	38671,8	59634,8	46411,9	2,3
Balanced Random	13,6	9,3	16243,5	48478,2	46027,7	2,8
Balanced Nearest	14,9	10,1	51588,1	55488,2	46230,7	2,4
Balanced Lateness	18,6	15,2	12191,9	49615,3	46101,26	4,20
Balanced Age	17,7	14,1	44161,0	47989,9	46020,5	4,7
Balanced Pile-on	17,6	14,0	12276,3	42349,2	45967,2	4,0

Pour évaluer l'interaction avec la règle POA, nous reproduisons l'expérience avec une allocation de commandes aléatoire (POA = *Random*) dans le TABLEAU 4.5.

Tableau 4.5 Comparaison des règles PPS (avec POA = Random)

Règle PPS	**OP**	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
Brute Force	**12,9**	**6,9**	**18830,7**	**19334,8**	**46606,8**	**2,2**
Random	12,3	4,54	33999,8	34723,7	46476,8	1,2
Balanced Random	11,8	3,94	36471,0	37206,0	46288,3	1,3
Balanced Nearest	11,3	3,8	32356,9	33112,9	46133,8	1,3
Balanced Lateness	12,6	5,7	29938,8	30516,3	46168,3	1,9
Balanced Age	14,0	6,5	27279,2	27796,9	45969,7	2,0
Balanced Pile-on	11,7	5,4	26577,0	27208,5	46131,8	2,0

Analyse : La règle *Brute Force* s'avère la plus performante. Avec POA = *Pod Matching*, elle affiche un OTHT de 10 140 s, soit environ 0,7 fois celui de la règle *Balanced Pile-on*. On observe également l'impact majeur du choix POA : pour la règle PPS *Brute Force*, l'OTHT est presque divisé par deux lorsqu'on passe d'un POA aléatoire (18 830 s) à *Pod Matching* (10 140 s).

4.2.3 Influence des règles de réapprovisionnement (ROA et RPS)

Nous testons ensuite l'impact des règles de réapprovisionnement, bien que la littérature les considère souvent comme secondaires.

Tableau 4.6 Comparaison des règles ROA (avec POA=*Pod Match*, PPS=*Brute Force*)

Règle ROA	**OP**	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
Random Cache	12,0	4,3	31213,3	31923,4	46452,8	1,2
Pod Batch	12,6	7,9	17752,4	66266,3	46344,1	2,3

Tableau 4.7 Comparaison des règles RPS (avec POA=*Pod Match*, PPS=*Brute Force*)

Règle RPS	**OP**	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
Random Item	13,3	8,8	17870,6	59634,8	46411,9	2,3
Turnover Item	**14,0**	**8,5**	**16172,0**	**62919,1**	**46318,1**	**2,3**
Closest Item	13,2	8,7	18161,3	67013,4	46276,7	2,1
Emptiest Item	14,0	7,9	18708,7	63694,4	46360,6	2,1
Least Demand	13,5	9,0	16817,1	59241,5	46354,7	2,3

Analyse : La règle ROA influence peu les performances. En revanche, pour le RPS, la règle

Turnover Item Storage (stockage basé sur la rotation des stocks) offre le meilleur OTHT, améliorant la performance d'environ 10% par rapport à d'autres stratégies.

4.2.4 Synthèse : Impact global de la règle POA

Enfin, nous comparons la configuration optimale (toutes les meilleures règles) avec la même configuration mais utilisant un POA aléatoire, afin de quantifier le gain spécifique du module POA.

Tableau 4.8 Performance configuration optimale (POA = *Pod Match*)

OP	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
15,9	11,2	12797,0	55879,7	46401,7	1,7

Tableau 4.9 Performance configuration dégradée (POA = *Random*)

OP	**OH**	**OTHT**	**OTOT**	**DT**	**IP**
13,8	6,8	17976,6	62232,6	46554,3	1,7

Les TABLEAUX 4.10, 4.11 et 4.12 résument le ratio des temps de traitement ($OTHT_{Random}/OTHT_{PodMatch}$) pour diverses combinaisons.

Tableau 4.10 Ratio OTHT (*Random/PodMatch*) selon la règle PPS

Règle PPS	**Ratio**
Brute Force	1,9
Random	0,9
Balanced Random	2,3
Balanced Nearest	0,6
Balanced Lateness	2,5
Balanced Age	0,6
Balanced Pile-on	2,2

Tableau 4.11 Ratio OTHT (*Random/PodMatch*) selon la règle ROA

Règle ROA	**Ratio**
Random	1,9
Same Pod	2,1

Tableau 4.12 Ratio OTHT (*Random/PodMatch*) selon la règle RPS

Règle RPS	**Ratio**
Random Item	1,9
Turnover Item	2,1
Closest Item	1,6
Emptiest Item	1,8
Least Demand	2,2

Conclusion : La majorité des ratios sont supérieurs à 1, confirmant que *Pod Matching* améliore significativement les délais. Les exceptions (ratios < 1 pour *Nearest* et *Age*) s'expliquent structurellement : ces règles sélectionnent les pods sur des critères purement géographiques ou temporels, ignorant la synergie de contenu que la règle POA tente de créer, annulant ainsi ses bénéfices.

La configuration **Baseline** retenue pour la suite est donc :

- POA : *Pod Matching*
- PPS : *Brute Force*
- ROA : *Pod Batch*
- RPS : *Turnover Item Storage*

4.3 Comparaisons des modèles Heuristique, MIP et MIPJ

Dans cette partie, nous utilisons les modèles linéaires représentés par MIP et MIPJ au sein de la simulation RAWSim-O. Nous allons d'une part évaluer l'efficacité de la prise de décision simultanée des problèmes POA et PPS par rapport à une prise de décision séquentielle prise par une heuristique H, qui correspond en fait à l'heuristique Pod Match. D'autre part nous comparerons nos deux modèles simultanés entre eux, pour desseller leurs différences et leur complémentarité.

Pour ce faire, nous avons choisi les paramètres exposés dans la partie 4.3.1, à partir de ceux choisis par les auteurs Jiao et al. [9], qui sont aussi les auteurs de l'article [10] d'où est tiré le modèle MIPJ. De même, nous avons choisi les autres règles de décision exposées dans la partie 4.3.2 à partir de ce même article.

4.3.1 Paramètres généraux des expériences

Tout d'abord, le TABLEAU 4.13 et la FIGURE 4.5 déterminent le nombre de répétitions effectuées pour chaque expérience, la durée de simulation et la configuration de l'entrepôt, utilisée dans nos expériences. La taille de ce dernier est beaucoup plus faible que celle de l'entrepôt décrit dans le TABLEAU 4.2 pour les expériences mettant en jeu des algorithmes heuristiques, car les modèles linéaires résolus par des solveurs sont très sensibles à la dimension du problème.

Tableau 4.13 Paramètres de la disposition de l'entrepôt

Paramètre	**Valeur**
Nombre de pods (dans les modèles : $\|\mathcal{P}\|$)	100
Nombre d'emplacements de stockage	120
Nombre de stations de sortie (dans les modèles : $\|\mathcal{S}\|$)	2
Nombre de stations de remplissage	2
Capacité des stations de sortie (dans les modèles : $\mathcal{C}_s$)	6
Temps de simulation	2 heures
Nombre de répétitions	10

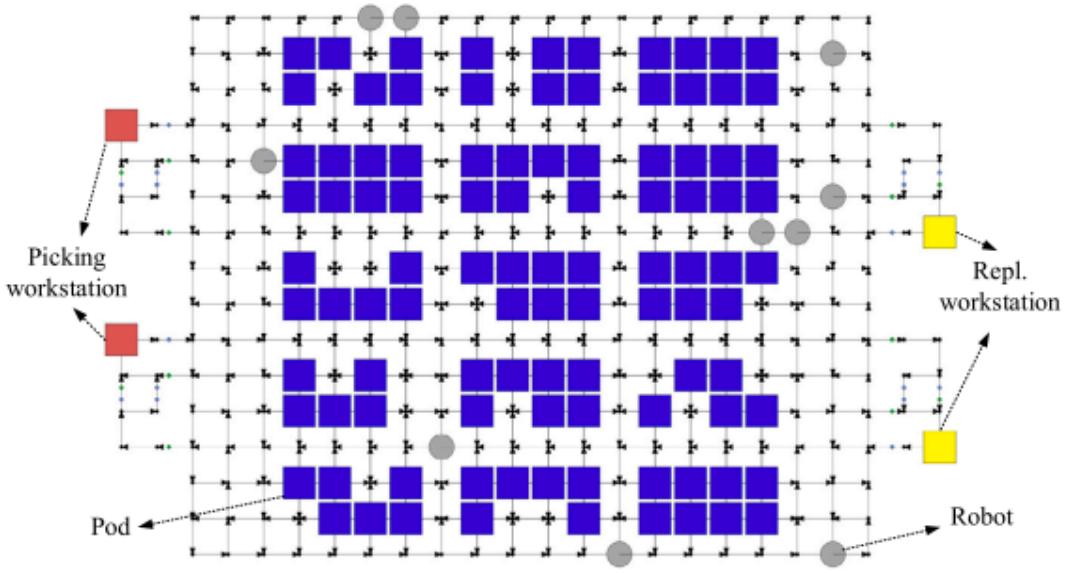

FIGURE 4.5 Disposition par défaut de l'entrepôt pour les expériences, décrite dans le TABLEAU 4.13, image provenant de l'article de Jiao et al [9]

Ensuite, nous présentons dans le TABLEAU 4.14 des paramètres présentant la configuration de l'entrepôt plus en détail. Ces paramètres permettent un travail sur un entrepôt très réaliste.

Tableau 4.14 Paramètres des différentes instances testées dans les expériences

Description	**Valeur**
Inventaire initial	70%
Capacité d'une étagère	100
Quantité de SKU différents	100
Taille d'une SKU	Uniforme entre 2 et 8
Fréquence SKU Quantité de lignes dans une commande	Normale 1,1, 1, 5
Quantité de SKU dans une commande	Normale 1,1, 1, 5
Taille du backlog de réapprovisionnement	variable
Quantité de lignes dans une commande de réapprovisionnement	Uniforme de 4 à 12
Temps pour mettre un item dans un pod	10s
Temps pour retirer un item d'un pod	3s
Temps pour gérer un item dans une station	10s
Capacité des stations de réapprovisionnement	1000
Accélération / décélération	1
Vélocité maximale	1,5s
Temps pour tourner	2,5s
Temps pour porter et déposer le pod	2,5s

4.3.2 Règle pour chaque problème de décision

Les autres paramètres définis correspondent aux différentes règles de décision pour les différents problèmes de décision. Nous avons basé le choix d'une telle combinaison de décision sur les résultats des expériences menées par Merschformann et al. [12]. Les définitions de ces règles sont données dans la liste des définitions. Les résultats qui avaient été obtenus par ces derniers sont montrés dans la FIGURE 4.6. La référence que l'on nomme H à laquelle les modèles linéaires sont comparés a été basée sur ces travaux plutôt que sur les expériences précédentes, car toutes les combinaisons de règles y ont été testées. Les différents problèmes de décision ayant une influence les uns sur les autres, tester les différentes règles de décision

sur une seule combinaison était moins représentatif.

Tableau 4.15 Paramètres des différentes instances testées dans les expériences

Problème de décision	**Règles H**	**Règles MIP et MIPJ**
PP	$WHCA^*_n$	$WHCA^*_n$
TA	Balanced	Balanced
POA	Pod Matching	MIP ou MIPJ
PPS	Nearest	MIP ou MIPJ
PR	Nearest	Nearest
ROA	Pod-Batch	Pod-Batch
RPS	Emptiest	Emptiest

RC rank	POA	ROA	PPS	RPS	PSA	performance
1	Pod-Match	Pod-Batch	Demand	Emptiest	Nearest	94.81%
2	Pod-Match	Pod-Batch	Demand	Emptiest	Station-Based	94.63%
3	Pod-Match	Pod-Batch	Nearest	Emptiest	Nearest	94.43%
4	Pod-Match	Pod-Batch	Demand	Emptiest	Class	94.00%

FIGURE 4.6 Combinaisons de règles de décision qui ont donné les meilleurs résultats en terme de throughput rate, tableau tiré de l'article de Merschformann et al. [12]

4.3.3 Description des expériences menées

Dans un premier temps, nous expérimentons différents déclencheurs ou *trigger* pour la résolution du modèle MIP au sein de la simulation afin de choisir celui permettant d'obtenir les meilleurs résultats. Il s'agit de définir l'instant auquel le modèle est de nouveau résolu pour envoyer de nouvelles décisions au simulateur. En particulier, cela sera lié au nombre de places disponibles au niveau des stations de travail. Nous utilisons ces données dans toutes les expériences suivant celle-ci.

Dans un second temps, nous reportons les quatre métriques importantes que sont la distance parcourue par les robots, le *throughput time*, l'*item pile-on* et les commandes effectuées pour différentes tailles de *backlog* de commandes pour les deux modèles d'intérêt et l'heuristique. Cela nous permet de tester l'hypothèse selon laquelle la prise de décision simultanée de nos deux problèmes d'intérêt permet d'accroître la synergie entre les commandes et les ressources de l'entrepôt en particulier lorsque la taille du *pool* initial de commandes augmente. De plus, cette expérience permet de tester et comparer l'adaptabilité de chaque modèle linéaire à la taille du *backlog*. Nous pourrons évaluer l'hypothèse selon laquelle laisser plus de flexibilité au modèle en ne reliant pas de manière définitive les *pods* avec les commandes permette une meilleure adaptabilité.

Dans un dernier temps, nous reportons de nouveau les quatre métriques importantes que sont la distance parcourue par les robots, le *throughput time*, l'*item pile-on* et les commandes effectuées pour un nombre de robots différent au sein de l'entrepôt pour nos trois modèles d'intérêt. Cette expérience permet de visualiser à partir de combien de robots dans l'entrepôt les ressources se trouvent saturées. Cela nous permettra également de tester l'hypothèse selon laquelle les modèles linéaires permettent une meilleure utilisation des robots et des étagères au sein de l'entrepôt par rapport au modèle H. Nous pourrons également évaluer de nouveau

l'adaptabilité de nos deux modèles MIP et MIPJ à la quantité de robots dans l'entrepôt et l'hypothèse selon laquelle le modèle le plus flexible, MIP, serait mieux adapté à un entrepôt contenant peu de robots comparé au modèle MIPJ.

le TABLEAU 4.16 présente de façon très synthétique les différents paramètres que l'on se propose de modifier lors des différentes expériences que nous menons dans le cadre de ce travail. En particulier, nous testerons quatre tailles différentes de backlog de commandes et sept quantités de robots se déplaçant au sein de l'entrepôt.

Tableau 4.16 Paramètres des différentes instances testées dans les expériences

Description	**Valeurs**
ΔDéclencheur	1,2,3 ou 4 commandes traitées avant d'appeler le modèle
ΔTaille du backlog avec 12 robots	25, 50, 100, 200
ΔNombre de robots avec backlog de 100 commandes	6, 7, 8, 9, 10, 11, 12

4.3.4 Choix du *trigger* ou déclencheur pour les modèles linéaires testés

La simulation nécessite un déclencheur permettant de lancer une nouvelle résolution du modèle pour de nouvelles allocations. Il s'agit d'attendre qu'un certain nombre de commandes soient effectuées au niveau de l'une ou l'autre des stations de travail, avant de lancer une nouvelle résolution.
Pour le modèle MIP, nous avons testé des valeurs allant de 1 à 4, ;ces résultats sont présentés sur la FIGURE 4.7.

Graphiquement, nous pouvons constater qu'en moyenne le nombre de commandes effectuées après avoir attendu qu'au moins une commande soit complétée dans l'une ou l'autre des stations de travail est d'environ 684 commandes, tandis que ce nombre s'élève toujours à moins de 656 commandes en moyenne pour les différents autres *triggers*. Bien que la donnée moyenne pour le *trigger* 2 ne soit pas significativement plus basse que celle obtenue pour le *trigger* 1, nous avons choisi d'effectuer nos expériences avec ce dernier.

Dans le cas du modèle MIPJ, nous avons suivi le déclencheur donné par les auteurs de l'article [10]. Il s'agit d'attendre que deux commandes soient complétées dans l'une ou l'autre des deux stations de l'entrepôt.

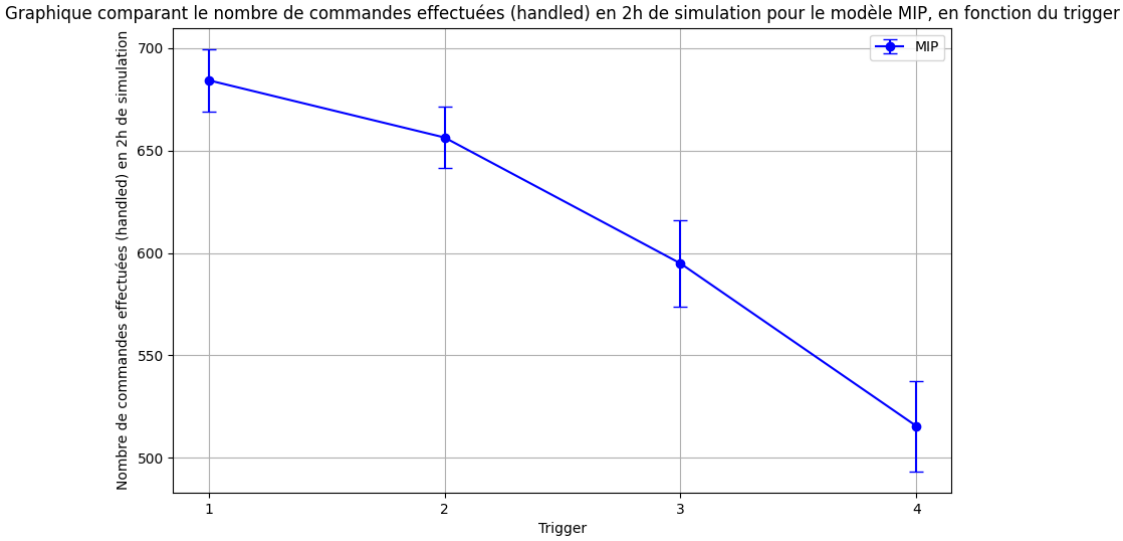

FIGURE 4.7 Nombre de commandes effectuées avec le modèle MIP pour différents déclencheurs

4.3.5 Variation de la taille du *backlog* de commandes

La seconde expérience menée consiste en la variation de la taille du *backlog* de commandes. Nous avons noté H le modèle heuristique, MIP le modèle développé dans le cadre de ce projet, dans la partie 3.2 et MIPJ le modèle tiré de l'article de Jiao et al. [10], correspondant au modèle 3.3. Nous souhaitons montrer qu'une augmentation de la taille du *backlog* causera une hausse de la synergie entre les deux problèmes de décision PPS et POA, résultant en une hausse des commandes traitées dans les cas MIPJ et MIP par rapport à celles traitées par le cas H. En effet, l'augmentation de la taille de backlog causera la hausse du nombre de commandes semblables ou demandant des items de même nature. Cette situation est plus sensible au groupement des commandes en lien avec l'inventaire disponible sur les étagères. Nous souhaitons également montrer que le modèle MIP présente une meilleure adaptabilité au nombre accru de commandes dans le *backlog* de l'entrepôt.

La FIGURE 4.8 consigne les résultats pour différentes métriques indiquant la performance de l'entrepôt pour les différents modèles. Nous avons moyenné les données sur dix répétitions. Les barres verticales représentent les écarts-types pour chaque série.
Toutes les résolutions ont atteint l'optimalité avec un gap d'optimalité de 0%.

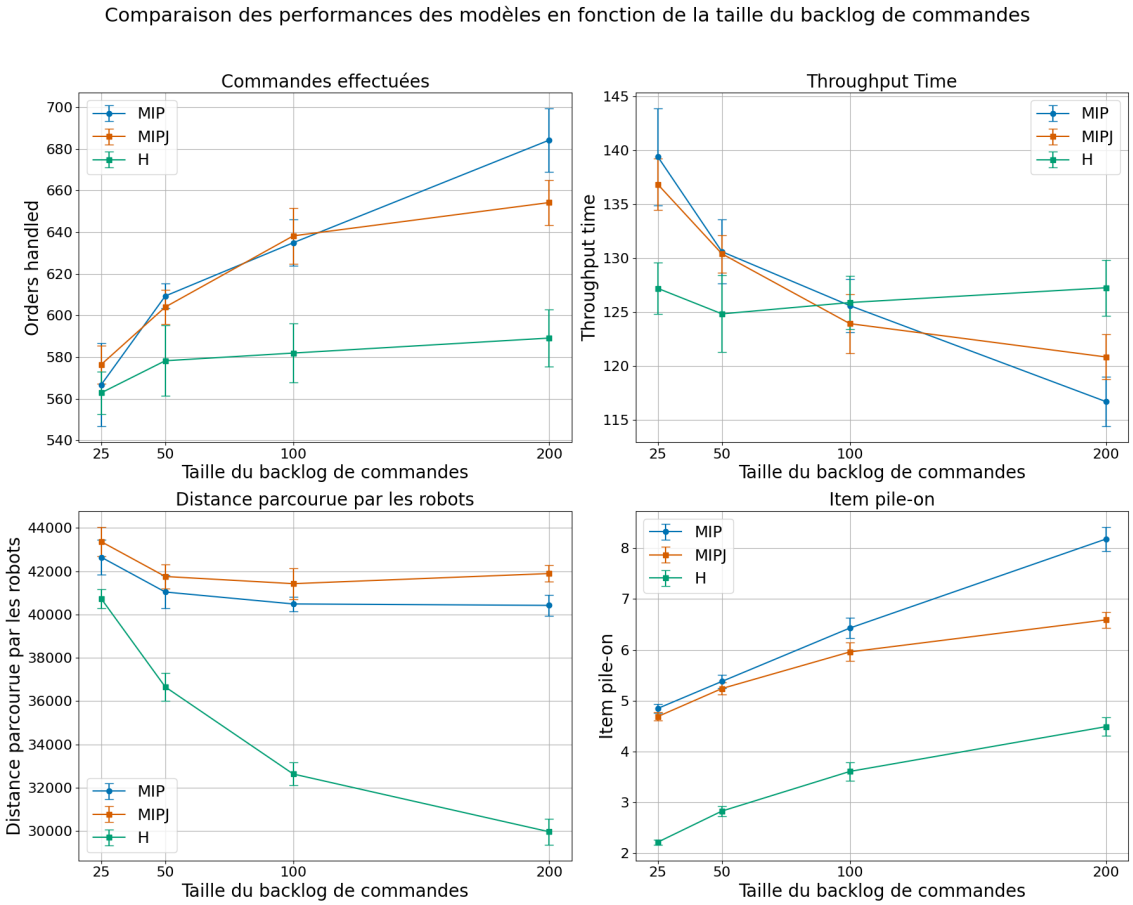

FIGURE 4.8 Courbes représentant le nombre de commandes traitées par les différents modèles, la distance parcourue par les robots, les item pile-on et les throughput times pour différentes tailles de backlog

Comparaison des modèles simultanés et séquentiel

En premier lieu, nous constatons que la performance de l'algorithme heuristique (H) plafonne malgré l'augmentation de la taille du *backlog*. Le nombre de commandes traitées n'augmente que très faiblement (facteur d'environ 1,05) lorsque l'on passe de 25 à 200 commandes. Cette stagnation se reflète logiquement sur le *throughput time*, métrique fortement corrélée au volume de commandes complétées. En revanche, l'algorithme H excelle dans l'optimisation des déplacements : la distance totale parcourue par les robots diminue de manière constante à chaque palier (facteur supérieur à 1,09), ceci étant à mettre en lien avec la règle régissant le choix des étagères pour cet algorithme, qui consiste à privilégier celles qui sont proches des stations. Parallèlement, le pile-on (nombre d'articles prélevés par étagère) s'améliore également (facteur $> 1{,}24$), bien que restant en deçà des modèles mathématiques.

Quant aux modèles linéaires (MIP et MIPJ), ils surclassent significativement l'approche heu-

ristique en termes de débit dès que le *backlog* atteint une taille critique de 50 commandes (meilleure performance d'un facteur de 1,04). Cet écart se creuse avec la charge, atteignant un facteur de 1,11 pour 200 commandes. Concernant les délais (*throughput times*), les modèles linéaires ne deviennent réellement plus performants que pour un *backlog* de 200 commandes (gain d'au moins 5 %). Pour des *backlogs* réduits (taille 25), le modèle H conserve l'avantage (rapidité supérieure d'un facteur 1,08), assurant une meilleure continuité de flux. Cependant, cette performance des modèles MIP se paie par une distance parcourue nettement plus élevée (jusqu'à 1,35 fois supérieure à celle du modèle H pour la taille 200). La véritable force des modèles linéaires réside dans leur capacité de groupement : leur taux de pile-on est systématiquement supérieur à celui du modèle H, avec un facteur multiplicatif minimum de 2,11.

Analyse : La réduction drastique de distance observée avec le modèle H s'explique par sa logique gloutonne : il privilégie les commandes dont les articles sont stockés dans des *pods* proches des stations. Avec un petit *backlog*, les opportunités sont rares, mais elles augmentent avec la taille du lot, améliorant mécaniquement le pile-on. Toutefois, cette heuristique reste myope : elle ignore des *pods* plus lointains qui permettraient pourtant de regrouper davantage de commandes. À l'inverse, les modèles linéaires acceptent de parcourir plus de distance pour maximiser le *pile-on*. Il apparaît donc que le gain de productivité offert par un meilleur groupement (approche simultanée) compense largement le coût logistique des trajets supplémentaires, rendant l'approche séquentielle (basée sur la distance) sous-optimale à grande échelle. Enfin, l'élargissement du *backlog* profite à tous les modèles en augmentant la diversité des articles et donc les opportunités d'optimisation.

Comparaison des modèles MIP et MIPJ

Les deux formulations mathématiques affichent des performances très proches en termes de commandes complétées, l'écart ne devenant significatif qu'avec un backlog de 200 commandes. Dans ce scénario de forte charge, le modèle MIP (flexible) devance le MIPJ (rigide) d'un facteur de 1,05, une tendance confirmée par une réduction équivalente du *throughput time*. La distance parcourue suit une courbe similaire pour les deux modèles, ne décroissant significativement qu'entre les tailles 25 et 200. Cependant, une distinction majeure apparaît sur le taux de *pile-on* : dès une taille de 100 commandes, le modèle MIP optimise mieux le groupement que le MIPJ (facteur 1,08), écart qui se creuse à 1,24 pour la taille 200.

Analyse : Bien que structurellement proches, le modèle MIP se distingue par sa meilleure adaptabilité aux grands volumes. Cette supériorité s'explique par sa flexibilité intrinsèque. Dans la simulation, l'allocation des ressources est soumise à des aléas (disponibilité des robots,

temps de trajet). Le modèle MIPJ, plus rigide, fige les associations commandes-*pods* trop tôt ; si un *pod* tarde à être déplacé, le système ne peut pas réallouer dynamiquement les commandes à d'autres *pods* plus opportuns. Le modèle MIP, en revanche, semble capable de "réviser" implicitement ses priorités, entraînant un meilleur pile-on et une distance parcourue moindre, optimisant ainsi le débit global.

4.3.6 Variation du nombre de robots

Cette expérience vise à évaluer la robustesse des modèles face à la variation des ressources logistiques (de 6 à 12 robots), pour un *backlog* fixé à 100 commandes. Nous anticipons logiquement une hausse du débit avec l'ajout de robots, jusqu'à un point de saturation imposé par la capacité finie des stations de sortie. L'enjeu est également de vérifier l'hypothèse selon laquelle le modèle flexible (MIP) gérerait mieux la pénurie de robots (scénarios à 6-8 robots) que le modèle rigide (MIPJ).

La FIGURE 4.9 présente l'évolution des métriques clés (commandes, distance, temps, pile-on) en fonction de la flotte disponible.

Comparaison des modèles simultanés et séquentiel

L'ajout de robots améliore le débit pour toutes les approches, mais un décrochage net s'opère en faveur des modèles linéaires dès 10 robots (surperformance de 7 % à 9 % par rapport à H). En termes de rapidité (*throughput time*), le modèle H reste plus efficace en situation de pénurie (6 à 8 robots), mais cet avantage s'érode à mesure que la flotte grandit. Mécaniquement, la distance totale parcourue augmente avec le nombre de robots actifs. Ici encore, les modèles linéaires induisent des trajets nettement plus longs que l'heuristique (facteur $> 1{,}11$). Le taux de pile-on, quant à lui, reste stable quel que soit le nombre de robots, mais demeure très largement supérieur pour les modèles linéaires (facteur $> 1{,}62$). Il est notable qu'au-delà de 9 robots, les gains de performance deviennent marginaux pour tous les modèles, signalant la saturation des stations de travail.

Analyse : La stagnation des résultats au-delà de 9 robots confirme que le goulot d'étranglement se déplace des robots vers les stations. Le modèle H confirme son profil "économe" en distance mais limité en capacité de groupement, ne parvenant pas à exploiter pleinement l'abondance de robots pour maximiser le débit, contrairement aux modèles MIP/MIPJ.

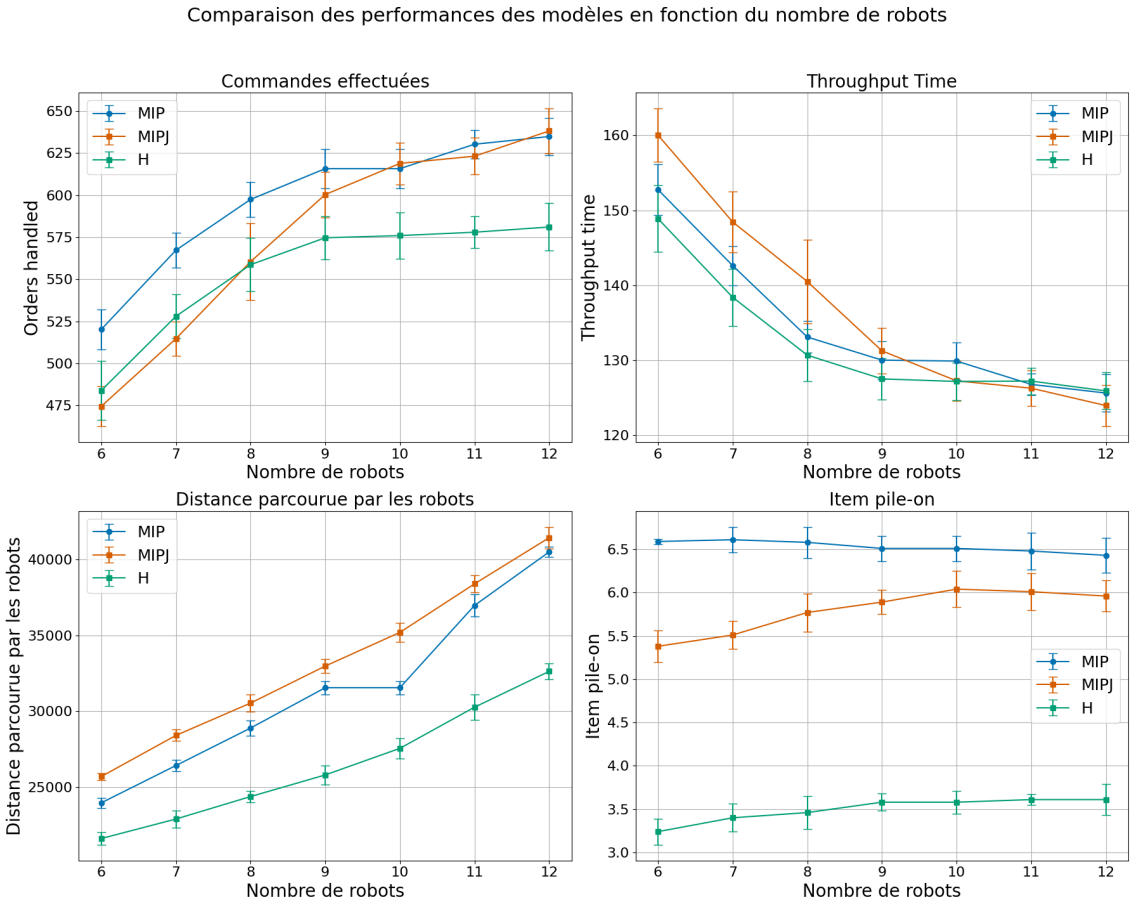

FIGURE 4.9 Évolution du nombre de commandes traitées selon la taille de la flotte de robots

Comparaison des modèles MIP et MIPJ

L'observation révèle une supériorité significative du modèle MIP en situation de ressources contraintes (6 à 8 robots). Avec 6 robots, le MIP traite environ 9 % de commandes en plus que le MIPJ, avec un *throughput time* réduit de 5 %. De plus, le MIP génère moins de déplacements inutiles (distance inférieure d'un facteur de 1,07) et maintient un *pile-on* systématiquement plus élevé (facteur $>$ 1,08). Bien que le MIPJ montre des temps de résolution plus rapides (facteur $>$ 1,42), sa rigidité le pénalise opérationnellement.

Analyse : Ces résultats valident notre hypothèse. Le modèle MIP est moins sensible à la raréfaction des robots. Sa flexibilité lui permet de mieux gérer les temps d'attente et l'indisponibilité des robots, là où le modèle MIPJ subit des blocages dus à ses décisions figées. Le MIP s'avère donc être l'approche la plus résiliente pour garantir la productivité dans un environnement aux ressources fluctuantes.

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

Ce mémoire a permis d'explorer le rôle crucial de l'automatisation logistique à l'ère du commerce électronique, en se focalisant spécifiquement sur les systèmes RMFS (*Robot Mobile Fulfillment Systems*). Nous avons modélisé cet environnement comme un système multi-agents complexe, régi par des contraintes d'interdépendance fortes, où l'allocation des ressources détermine directement la performance opérationnelle. Notre étude s'est concentrée sur l'optimisation conjointe des deux problèmes décisionnels les plus critiques : l'affectation des commandes aux stations (*Pick Order Assignment - POA*) et la sélection des étagères (*Pick Pod Selection - PPS*).

La contribution majeure de ce travail réside dans la formulation et la comparaison de modèles de programmation linéaire en nombres entiers, conçus pour résoudre ces problèmes de manière simultanée plutôt que séquentielle. L'implémentation de ces modèles au sein d'un framework de simulation réaliste et dynamique a permis de valider nos hypothèses dans des conditions proches du réel.

Les résultats expérimentaux démontrent sans équivoque la supériorité de l'approche intégrée. La prise de décision simultanée permet de mieux exploiter les synergies entre les commandes et la disponibilité des stocks, générant un débit (*throughput*) supérieur aux meilleures heuristiques séquentielles de la littérature. Plus spécifiquement, le modèle "Flexible" développé dans le cadre de cette recherche s'est distingué par sa robustesse. Contrairement au modèle "Rigide" , sa capacité à ne pas figer prématurément les liens entre commandes et étagères lui confère une meilleure adaptabilité, particulièrement dans les scénarios de forte charge ou de pénurie de robots.

5.2 Limitations

Malgré ces résultats prometteurs, notre approche présente des limitations inhérentes à la nature des méthodes exactes. La complexité calculatoire des modèles linéaires restreint leur applicabilité à des instances de taille modeste ou nécessite des temps de calcul incompatibles avec les exigences du temps réel strict dans des entrepôts de très grande envergure. Ainsi, bien que ces modèles fournissent des solutions optimales ou quasi-optimales, ils agissent davantage comme une preuve de concept et une borne supérieure de performance que comme une solution immédiatement déployable à grande échelle industrielle.

5.3 Améliorations futures

Ces limitations ouvrent la voie à plusieurs pistes de recherche prometteuses pour prolonger ces travaux. Une voie de recherche en plein essor consiste à utiliser des agents basés sur le *Deep Reinforcement Learning* comme cela a été fait dans un article de Rimélé [15] pour le problème de rangement des étagères après avoir été utilisées au niveau des stations de travail. Ces agents pourraient apprendre des stratégies d'allocation complexes en observant les résultats de la simulation sur le long terme, capturant ainsi des dynamiques subtiles que la programmation mathématique statique peine à anticiper.

Enfin, il serait pertinent d'élargir le modèle simultané à d'autres décisions critiques du système RMFS, telles que la gestion de la congestion ou la gestion de l'énergie (avec l'allocation des bornes de recharge). Une approche intégrant ces contraintes pourrait encore affiner la précision et le réalisme opérationnels ainsi que la productivité globale du système.

RÉFÉRENCES

[1] Kaveh Azadeh, René De Koster, and Debjit Roy. Robotized warehouse systems : Developments and research opportunities. *ERIM report series research in management Erasmus Research Institute of Management*, (ERS-2017-009-LIS), 2017.

[2] John J Bartholdi and Steven T Hackman. Warehouse & distribution science : release 0.96. *The supply chain and logistics institute*, 30332, 2014.

[3] Nils Boysen, Dirk Briskorn, and Simon Emde. Parts-to-picker based order processing in a rack-moving mobile robots environment. *European Journal of Operational Research*, 262(2) :550–562, 2017.

[4] Nils Boysen, Konrad Stephan, and Felix Weidinger. Efficient order consolidation in warehouses : The product-to-order-assignment problem in warehouses with sortation systems. *IISE Transactions*, 54(10) :963–975, 2022.

[5] Tolga Cezik, Stephen C Graves, and Amy C Liu. Velocity-based stowage policy for a semiautomated fulfillment system. *Production and Operations Management*, 2022.

[6] René de Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking : A literature review. *European Journal of Operational Research*, 182 :481–501, 2007.

[7] John Enright and Peter R Wurman. Optimization and coordinated autonomy in mobile fulfillment systems. In *Automated action planning for autonomous mobile robots*, pages 33–38, 2011.

[8] Michael Hompel and Thorsten Schmidt. *Warehouse management : automation and organisation of warehouse and order picking systems*. Springer Science & Business Media, 2006.

[9] Guoshuai Jiao, Min Huang, Yang Song, Haobin Li, and Xingwei Wang. Online joint optimization of order picking process in robotic mobile fulfillment systems. *Omega*, 138 :103374, 2026.

[10] Guoshuai Jiao, Haobin Li, and Min Huang. Online joint optimization of pick order assignment and pick pod selection in robotic mobile fulfillment systems. *Computers & Industrial Engineering*, 175 :108856, 2023.

[11] Tim Lamballais, Debjit Roy, and MBM De Koster. Estimating performance in a robotic mobile fulfillment system. *European Journal of Operational Research*, 256(3) :976–990, 2017.

[12] Marius Merschformann, Tim Lamballais, MBM De Koster, and Leena Suhl. Decision rules for robotic mobile fulfillment systems. *Operations Research Perspectives*, 6 :100128, 2019.

[13] Marius Merschformann, Lin Xie, and Daniel Erdmann. Path planning for robotic mobile fulfillment systems. *arXiv preprint arXiv :1706.09347*, 2017.

[14] Marius Merschformann, Lin Xie, and Hanyi Li. Rawsim-o : A simulation framework for robotic mobile fulfillment systems. *arXiv preprint arXiv :1710.04726*, 2017.

[15] Adrien Rimélé, Philippe Grangier, Michel Gamache, Michel Gendreau, and Louis-Martin Rousseau. E-commerce warehousing : learning a storage policy. *arXiv preprint arXiv :2101.08828*, 2021.

[16] Kees Jan Roodbergen and Iris FA Vis. A survey of literature on automated storage and retrieval systems. *European journal of operational research*, 194(2) :343–362, 2009.

[17] U.S. Census Bureau. E-commerce activity across sectors (2019-2020), 2025. Consulté le 26 Nov. 2025. [En ligne] : `https://www.census.gov/library/visualizations/interactive/e-commerce-activity-across-sectors-2019-2020.html`.

[18] Cristiano Arbex Valle and John E Beasley. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. *Computers & Operations Research*, 125 :105090, 2021.

[19] Bingqian Wang, Xiuqing Yang, and Mingyao Qi. Order and rack sequencing in a robotic mobile fulfillment system with multiple picking stations. *Flexible Services and Manufacturing Journal*, pages 1–39, 2022.

[20] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1) :9–9, 2008.

[21] Lin Xie, Hanyi Li, and Nils Thieme. From simulation to real-world robotic mobile fulfillment systems. *arXiv preprint arXiv :1810.03643*, 2018.

[22] Lin Xie, Nils Thieme, Ruslan Krenzler, and Hanyi Li. Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. *European Journal of Operational Research*, 288(1) :80–97, 2021.

[23] Xiying Yang, Guowei Hua, Li Zhang, TC Cheng, and Tsan Ming Choi. Joint order assignment and picking station scheduling in kiva warehouses with multiple stations. *arXiv preprint arXiv :2108.09056*, 2021.

ANNEXE A DÉTAILS D'IMPLÉMENTATION ET MODIFICATIONS DU FRAMEWORK RAWSIM-O

Cette annexe a pour vocation de détailler les structures logicielles régissant le fonctionnement de la simulation, spécifiquement au niveau des problèmes de décision identifiés à la FIGURE 4.3. Nous y décrivons les adaptations architecturales réalisées au sein du framework pour permettre l'intégration de notre règle de décision conjointe (POA + PPS) basée sur la programmation linéaire. Pour faciliter la compréhension de ces mécanismes, nous nous appuyons sur le diagramme de classes présenté en FIGURE A.1.

A.1 Architecture logicielle du processus de décision

Le diagramme de classes (FIGURE A.1) ne couvre pas l'intégralité de la simulation, mais se concentre sur les composants impliqués dans les décisions d'allocation (POA et PPS), ainsi que la décision d'affectation des tâches (TA), intrinsèquement liée au PPS par l'architecture du framework (voir section 4.1.2).

Au cœur du dispositif se trouve la classe `Controller`. C'est le chef d'orchestre de la simulation : elle contient la boucle principale (`Update`) qui gère l'avancement du temps et déclenche séquentiellement la mise à jour de tous les agents.

A.1.1 Phase de Planification : Le rôle de l'OrderManager

Le processus débute par la mise à jour du gestionnaire de commandes, représenté par la classe `OrderManager`. Cette classe, qui implémente l'interface `IUpdateable`, possède sa propre méthode de mise à jour déclenchée par l'arrivée de nouvelles commandes dans le backlog (attribut `PendingOrder`).

Dans l'architecture native de RawSim-O, la méthode `DecideAboutPendingOrder` traite les décisions de manière séquentielle et heuristique. Dans le cadre de notre projet, nous avons modifié ce comportement pour intégrer la prise de décision conjointe :

- L'appel de la méthode déclenche désormais l'exécution de la classe `MIP`.
- Cette nouvelle classe encapsule la logique d'optimisation : elle construit et résout les modèles linéaires via la fonction `Solve`.
- Alors que l'assignation des commandes aux stations (POA) est appliquée directement à l'issue du calcul, l'allocation des étagères (PPS) doit être différée pour être synchro-

nisée avec la disponibilité des robots.

— Pour gérer ce délai, la solution optimale concernant les étagères est stockée dans une structure de données intermédiaire : le dictionnaire `Xps`, défini au sein de la classe `ResourceManager`.

A.1.2 Phase d'Exécution : Interaction Robot et BotManager

La concrétisation opérationnelle des décisions prises par le modèle MIP s'effectue lors de la mise à jour des agents robots (classe `BotNormal`). Lorsqu'un robot est disponible, sa méthode `Act` (héritée de l'interface `IBotState`) interroge le `BotManager` via la fonction `GetNextTask` pour obtenir une nouvelle mission.

C'est ici qu'intervient le couplage avec notre modèle :

— Lorsqu'une tâche d'extraction est requise, la fonction `DoExtractTaskForStation` de la classe `BotManagerPodSelection` est sollicitée.

— Nativement, cette fonction sélectionnait un pod "à la volée" selon des critères géographiques.

— Dans notre implémentation, cette fonction a été réécrite pour consulter le dictionnaire `Xps` du `ResourceManager`. Elle filtre ainsi les pods éligibles pour ne retenir que ceux qui ont été explicitement désignés par la solution optimale du modèle linéaire.

Une fois le pod validé, les requêtes liant ce pod aux commandes de la station sont enregistrées via la fonction `EnqueueExtract`, garantissant que le robot exécutera précisément le plan optimisé.

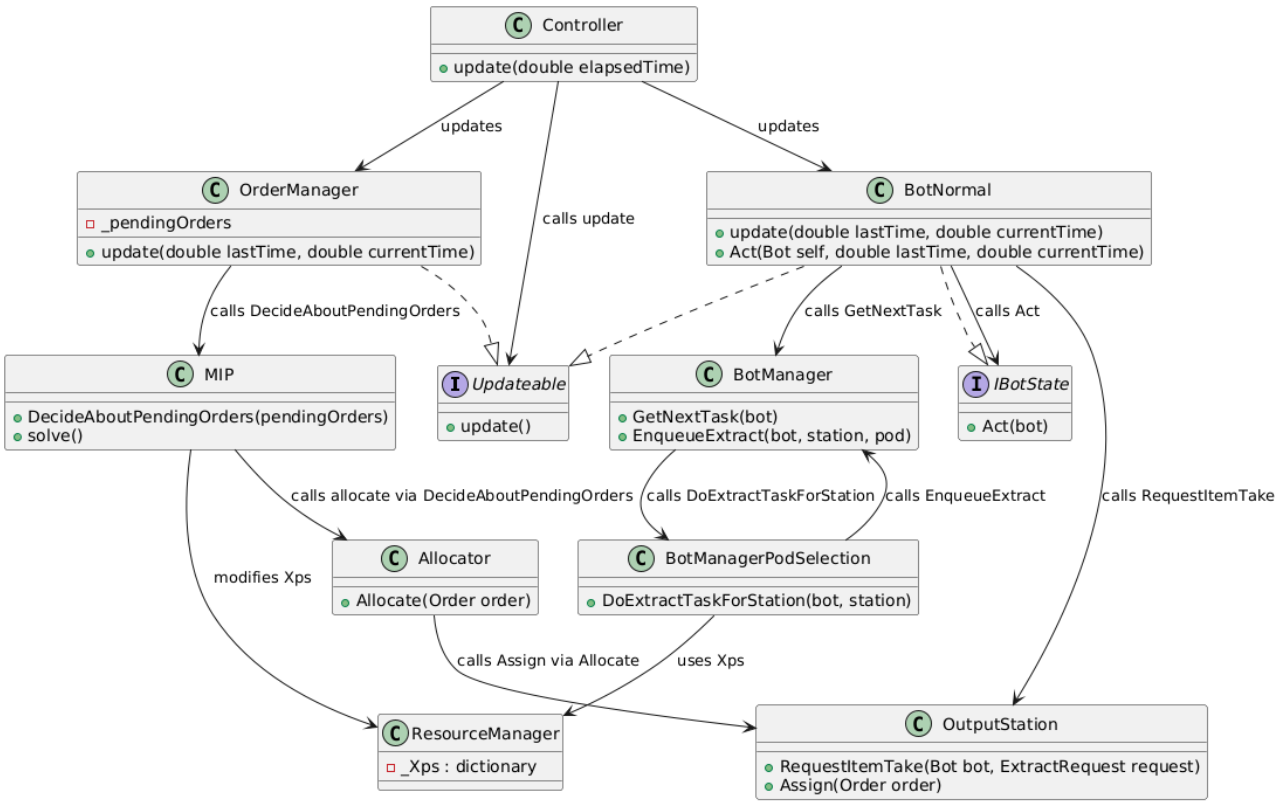

FIGURE A.1 Diagramme de classes de notre processus de décision, intégré dans la simulation d'origine RawSim-O

A.2 Synthèse des modifications apportées

La FIGURE A.2 met en évidence les quatre interventions majeures effectuées sur le code source du framework pour permettre cette intégration :

— **Modification 1 (Nouvelle Classe) :** Ajout de la classe `MIP`. Ce module est responsable de la génération des variables, des contraintes et de la résolution du modèle mathématique.

— **Modification 2 (Logique de Décision) :** Modification de la méthode `DecideAboutPendingOrder` dans l'`OrderManager` pour dérouter le flux de décision standard vers notre classe `MIP`.

— **Modification 3 (Persistance des Données) :** Ajout de l'attribut public `Xps` (dictionnaire) dans `ResourceManager`. Il sert de mémoire tampon partagée pour transmettre les résultats de l'allocation des pods aux robots.

— **Modification 4 (Filtrage des Tâches) :** Réécriture de la méthode `DoExtractTaskForStation` dans `BotManagerPodSelection` pour forcer la sélection

des pods selon les directives du dictionnaire `Xps`, remplaçant ainsi la logique gloutonne par défaut.

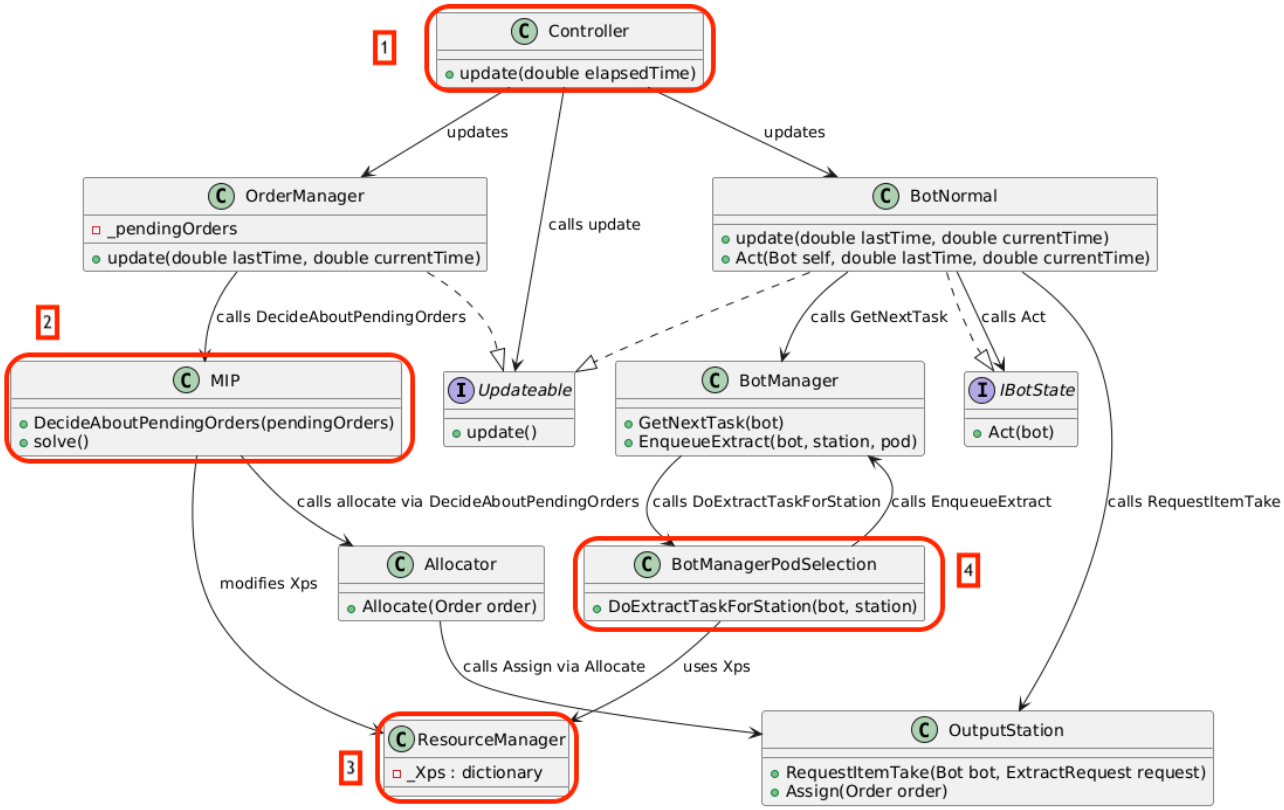

FIGURE A.2 Diagramme de classes de notre processus de décision, intégré dans la simulation d'origine RawSim-O avec ses principales modifications

Enfin, il convient de rappeler que ces mécanismes opèrent dans un environnement sous contrainte constante : le backlog de commandes est maintenu à un niveau fixe (paramétrable par l'utilisateur), chaque commande assignée étant immédiatement remplacée par une nouvelle entrée, assurant une pression continue sur les algorithmes d'optimisation.