Master Systèmes Dynamiques et Signaux

Mémoire

Modélisation dynamique et optimisation de flux logistiques appliquées à une installation robotisée par Balyo.

Auteurs :
M^me^ Carla Juvin

Jury :
Pr. Laurent Hardouin
Dr. Nicolas Delanoue
Dr. Remy Guyonneau
Pr. Jean-Baptiste Fasquel
Pr. David Rousseau

Version du
27 juin 2020

Remerciements

Je tiens à remercier l'entreprise Balyo de me permettre de travailler sur ce projet, notamment les membres du projet TESSERACT, Thomas BARBILLON, Guillaume MATIVON et Basile LACOMBE BAR. Je tiens à remercier particulièrement Aurélien GOSSEYE qui encadre ce projet.

Je souhaite aussi remercier Nicolas DELANOUE pour ses conseils et le soutien qu'il m'a apporté durant toutes les phases de ce projet. Je remercie aussi Sébastien LAGRANGE pour son aide.

Table des matières

Table des figures

Liste des acronymes

ASP	*Answer Set Programming*
CBM	*Conflict Based Min-cost-flow*
ICBS	*Improved Conflict-BasedSearch*
MAPF	*Multi-Agent Path Finding*
MPP	*Multi-robot Path Planning*
PDP	*Pickup and Delivery Problem*
TAPF	*combined Target Assigment and Path Finding*
VRP	*Vehicle Routing Problem*
VRPTW	*Vehicle Routing Problem with Time Windows*
WMS	*Warehouse Management System*

Chapitre 1

Introduction

Ce document présente les recherches effectuées dans le cadre du master SDS en partenariat avec l'entreprise Balyo. Il concerne principalement des problèmes de tournées de véhicules électriques avec des contraintes de gestion efficace des batteries et de leur recharge. La section 1.1 présente le projet dans sa globalité. On décrit les éléments qui composent une installation dans la section 1.2. Finalement, l'objectif de mon travail est expliqué dans la section 1.3.

1.1 Présentation du projet

Ce partenariat s'inscrit dans une logique de déploiement d'une solution intelligente pour l'industrie 4.0. Le projet Tesseract a pour objectif la navigation autonome des chariots de manutention robotisés en entrepôt.

Le rôle de ces chariots robotisés est de répondre à des missions qui consistent à déplacer des marchandises dans l'entrepôt.

L'objectif de ce projet est d'optimiser le trafic en améliorant la génération de trajectoires afin de limiter les temps de trajets, respecter les aires de circulation autorisées et éviter les obstacles.

Le projet comprend d'ores et déjà des solutions permettant de :
- définir les zones de circulation, par des routes, sur la carte d'un entrepôt,
- détecter les intersections entre ces routes pour générer un graphe de l'ensemble des chemins possibles,
- trouver le meilleur chemin dans ce graphe,
- trouver le meilleur itinéraire (les positions exactes du robot) le long de ce chemin.

L'avancement du projet permet donc dès maintenant de dessiner facilement les zones de circulation et de générer des trajectoires fluides et précises pour les robots. Cependant,

certains aspects restent à améliorer, notamment la gestion du trafic, c'est sur cette question que porte ce travail. La suite de ce chapitre expose les éléments et les caractéristiques d'une installation classique d'un client de Balyo.

1.2 Présentation d'une installation

Le client dispose
— d'une flotte de n robots,
— d'un entrepôt $\mathcal{E}$ avec sa géométrie.
La Figure 1.1 donne une illustration d'une installation robotisée.

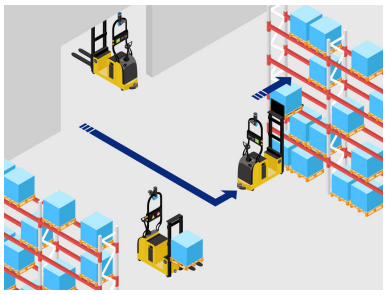

FIGURE 1.1 – Installation robotisé par Balyo

1.2.1 Les robots

Chaque robot est caractérisé par :
— sa géométrie (son gabarit),
— sa capacité à effectuer une tâche : atteindre une hauteur, supporter une charge, ...
Un robot est caractérisé, dynamiquement, par :
— sa position dans l'espace (x, y, θ),
— la mission m qu'il est en train de réaliser,
— son niveau de batterie disponible,
— le nombre de missions déjà réalisées,
Un exemple de robot est donné par Figure 1.2.

1.2.2 Un entrepôt

La carte est une représentation géométrique de l'entrepôt. Celle-ci est générée via un algorithme de localisation et cartographie simultanées, Simultaneous Localization And Mapping (SLAM). Un exemple de carte est donné par l'illustration 1.3.

FIGURE 1.2 – Chariot Autonome robotisé par Balyo

Sur cette carte, la solution TESSERACT consiste tout d'abord à placer des objets vectoriels, afin de paramétrer la géométrie du circuit. Plus précisément, on trouvera :

— des tronçons qui vont décrire les règles de circulation et les zones où les robots sont autorisés à circuler,
— des points d'intérêts.

Pour le plan géométrique 1.3, la figure 1.4 donne un exemple de plan vectoriel.

1.2.3 Les tronçons

Plus formellement, chaque tronçon est caractérisé par

— les coordonnées de ses sommets,
— son orientation,
— une règle de circulation (sens unique ou double sens),
— une marge de sécurité.

Cet ensemble de tronçons forme le circuit. A partir de ce circuit, et d'un couple de nœuds (de type arrivée, départ), on peut générer une trajectoire que devra suivre le robot.

1.2.4 Les points d'intérêt

Les points d'intérêt peuvent correspondre à différents besoins :

— points de prise et de pose (pour les missions),
— stations de recharge de batterie,
— zones taxis

On note que les points de prises et de poses sont souvent de l'un de ces deux types : emplacement de stockage dans l'entrepôt ou bien zone de dépose suite à une livraison d'un camion.

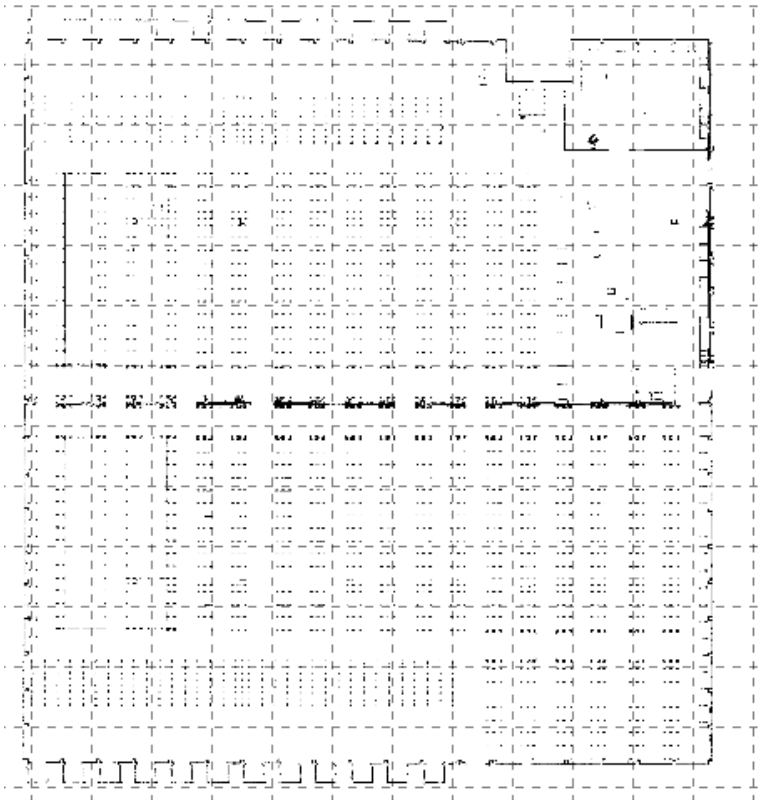

FIGURE 1.3 – Exemple de plan d'un entrepôt.

Formellement, un point d'intérêt est caractérisé par sa position (x, y) et son orientation θ.

1.2.5 Les missions

La plupart du temps, le client dispose d'un WMS (Warehouse Management System) qui sert à l'organisation des entrepôts (gestions des stocks, utilisation des surfaces de stockages, ...).

Plus précisément, le WMS fournit une liste de missions à réaliser. Chaque mission est composée de :

- un horaire t auquel cette mission est tombée,
- un point de prise P (pickup point),
- un point de pose D (delivery point),
- une contrainte de priorité qui peut être de différentes natures : priorité, une échéance, takt time (durée idéale de la mission).

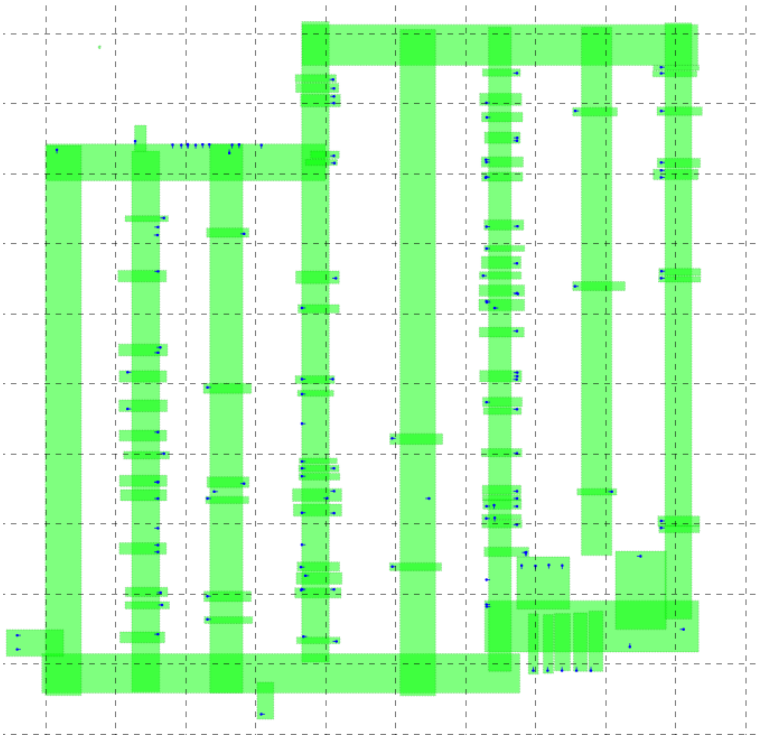

FIGURE 1.4 – Exemple de plan TESSERACT d'un entrepôt.

1.2.6 Les perturbations

Les perturbations suivantes peuvent engendrer des embouteillages ou une perte de temps :

- obstacles sur une chemin, (e.g. robot en erreur, palette déposée sur le sol au milieu du passage, autre cariste ...)
- palettes absentes au point de prise,
- emplacement de pose non disponibles.

1.3 Objectif du projet

L'objectif de ce projet est de proposer un ordonnanceur capable d'affecter dans le temps, les missions aux robots.

La suite de ce document est organisée de la façon suivante. Le chapitre 2 de ce document donne des références bibliographiques pouvant servir se support pour l'affectation des missions à une flotte de robots (Ordonnancement) voir section 2.1. Le chapitre 3 présente le travail réalisé dans le cadre du stage du master. Enfin le chapitre 4 détaille les premiers résultats obtenus.

Chapitre 2

Étude bibliographique

Ce chapitre regroupe l'ensemble des recherches bibliographiques menées lors de ce projet concernant les problèmes d'ordonnancement, plus particulièrement les problèmes de tournées de véhicules.

Lors de l'élaboration de la bibliographie, on s'était aussi intéressé à la littérature se rapportant à la planification de chemins pour plusieurs agents ainsi qu'aux méthodes existantes pour la résolution combinée des deux problèmes précédemment décrits. Finalement, ces sujets n'ont pas été abordés dans la suite du travail, ils ne sont donc pas reportés dans ce chapitre.

2.1 Ordonnancement des tâches

Dans ce chapitre, nous allons nous intéresser à la répartition des missions entre les robots et leurs enchaînements. Il s'agit alors d'un problème d'ordonnancement : sélectionner les ressources pour que les opérations soient réalisées de façon optimale.

Définition 2.1 (Ordonnacement)
L'ordonnancement consiste à organiser dans le temps la réalisation d'un ensemble de tâches, en prenant en compte ses contraintes :
- *Temporelles (délais, priorité, ...),*
- *De ressources (capacité, disponibilité ...),*

pour atteindre un objectif (minimiser les coûts, maximiser la production ...).

Pour cela, nous nous sommes intéressés aux problèmes de tournées de véhicules (aussi appelés VRP pour Vehicle Routing Problem).

2.1.1 Définition du problème

VRP est une classe de problèmes de recherche opérationnelle qui permet de déterminer les tournées d'une flotte de véhicules afin de réaliser des missions (livraison, visites, ...) tout en minimisant le coût.

Le problème de tournées de véhicules est NP-Complet [8], ainsi, l'espace de recherche croit de façon exponentielle avec la dimension du problème. Il n'existe pas d'algorithme efficace connu permettant de trouver les solutions optimales. Les métaheuristiques permettent alors d'explorer une partie de ces solutions en se concentrant sur celles qui paressent les plus prometteuses, ce qui permet de réduire le temps de calcul et obtenir une solution sous-optimale mais acceptable.

Il existe de nombreuses variantes du problème de tournées de véhicules [14], les suivantes correspondent, en partie, à notre problème :

— VRP avec contrainte de capacité : CVRP
— VRP avec fenêtres temporelles : VRPTW, chaque client dispose d'une fenêtre temporelle durant laquelle il peut être livré.
— VRP avec ramassage et livraison : des entités doivent être déplacées d'une position d'origine à une position de destination.
— VRP avec véhicules électriques : eVRP

En effet, chaque robot ne peut transporter qu'un nombre limité de palettes à la fois, la capacité de celui-ci est donc limitée. D'autre part, il est possible que les missions doivent être réalisées avant une *deadline* donnée, d'où la nécessité d'introduire des fenêtres temporelles. De plus, notre problème est exactement un problème de ramassage et de livraison car les missions des robots consistent à déplacer les palettes d'un point de ramassage à un point de livraison. Enfin, les robots fonctionnent à l'électricité et cela entraine des contraintes particulières liées à l'autonomie des véhicules.

Dans la suite du chapitre, on s'intéressera en particulier au VRP avec ramassage et livraison et au VRP avec véhicules électriques.

2.2 VRP avec ramassage et livraison

Chaque transport doit lier une origine et une destination précise. On parle alors de Pickup and Delivery Problem (PDP).

Dans la version standard du PDP, les missions sont effectuées par une flotte de véhicules identiques, stockés dans un même dépôt. Le but est planifier un ensemble de routes, capable de servir toutes les missions en respectant un ensemble de contraintes et en minimisant les coûts de transport.

Représentation du problème par un graphe

Cordeau [2] formule le problème de pickup and delivery à l'aide d'un graphe $G = (V, A)$. L'ensemble des $2n + 1$ nœuds V est le suivant $\{0, P_1, \ldots, P_N, D_1, \ldots, D_N, 2n + 1\}$ de sorte que les nœuds 0 et $2n + 1$ représentent deux copies du dépôt, les noeuds $P = \{P_1, \ldots, P_N\}$ représentent les points de prise des N missions et les noeuds $D = \{D_1, \ldots, D_N\}$ représente les points de livraison des N missions. Une mission est un couple $(i, i + N)$, avec $i \in P$ et $i + N \in D$. Deux contraintes sont donc mises en œuvre dans ce genre de problème :

— pour chaque demande de transport, le nœud de prise i doit être visité avant celui de pose $i + N$,

— le nœud de prise i et de pose $i + N$ d'une même demande de transport sont visités dans la même tournée, par le même véhicule.

L'objectif est donc de trouver pour chaque robot un parcours dans ce graphe de sorte que, chaque nœud de P et de D soit visité exactement une fois.

2.2.1 Approches exactes

Psaraftis [13] a formulé et résolu le problème, PDP, avec un unique véhicule dès 1980. La plus grand instance résolue était alors composée de neuf demandes. En 2006, Cordeau [2] a introduit une formulation du problème en programmation linéaire en nombres entiers, qui, combinée à un algorithme de branch-and-cut, a permis de résoudre des instances allant jusqu'à 32 demandes. Savelsbergh et Sol (1998)[16] ont développé un autre algorithme exact pour le PDP avec fenêtres de temps, qui est capable d'insérer de nouvelles demandes de façon dynamique. Une synthèse de plusieurs algorithmes exactes pour le PDP est présentée dans le Tableau 2.1

reference	hypothèse	nb de missions
[13], [12]	1 seul véhicule	≤ 9
[17]	1 seul véhicule	≤ 18
[16]	dynamique	≤ 30
[3]	fenêtres de temps	≤ 40
[2]	fenêtres de temps	≤ 32
[15]	fenêtres de temps	≤ 96

TABLE 2.1 – Synthèse de plusieurs algorithmes exactes pour le PDP

2.2.2 Heuristiques

Kouki et al. [7] et Hanif et al. [5] proposent des solutions basées sur la négociation entre les agents. Les tâches sont alors affectées une à une, pour chacune d'entre elles un agent est choisi parmi ceux disponibles.
Les articles [11][9] et [6] utilisent des méthodes d'optimisation métaheuristiques telles que la recherche avec tabous ou la méthode du recuit simulé. Une synthèse de plusieurs algorithmes approchés pour le PDP est présentée dans le Tableau 2.2

reference	hypothèse	modèle
[7]	requètes arrivent 1 par 1	négotiations traitées 1 par 1, insertion nouvelles requètes et re-optimisation
[5]	deux types d'agents, colis et camions qui disposent d'informations et peuvent agir	Un AGV est responsable de trouver les colis à proximité et de choisir un chemin d'accès.
[11][9]	Plusieurs véhicules fenêtres de temps	recherche avec méthode tabou
[6]	Plusieurs véhicules fenêtres de temps	recuit simulé

TABLE 2.2 – Synthèse de plusieurs algorithmes approchés pour le PDP

2.3 Véhicules électriques

La logistique des véhicules électriques ajoute une difficulté supplémentaire aux problèmes classiques de tournées de véhicules, en imposant de nouvelles contraintes. En effet, ces véhicules ayant une autonomie limitée, ils doivent se rendre aux stations de recharge en cours de tournée. Il n'est donc pas possible d'utiliser naïvement les techniques de base pour ces problèmes.

2.3.1 Approches exactes

Il existe des méthodes exactes de résolution utilisant la programmation mathématique, inspirées de la formulation du green vehicle routing problem(GVRP) proposé par Erdogan et Miller-Hooks [1].

Cependant, à cause de leur complexité, il ne peuvent être utilisé que sur des instances de taille limitée. C'est pourquoi des méthodes approchées ont aussi été développées.

2.3.2 Heuristiques

Montoya [10] est parvenu à résoudre des instances de taille industrielle en utilisant des méthodes de recherche locale itérative. Froger et al. [4] ont en plus pris en compte des contraintes de capacité pour les stations de recharge, contraintes qui s'appliquent aussi pour les problèmes de chariots automatisés auxquels on s'intéresse.

2.4 Conclusion

Dans ce chapitre une étude bibliographique nous a permis de déterminer les solutions existantes et les termes utilisés dans la communauté scientifique pour des problèmes comparables à celui posé par Balyo. Ainsi, nous avons pu rapprocher le problème d'affectation des tâches à un *Pickup and Delivery Problem* et à un *Electric Vehicle Routing Problems*. Ces résultats ont ensuite permis de proposer des implémentations adaptées, présentées dans le chapitre suivant.

Chapitre 3

Travaux réalisés

L'objectif principal de ce projet est de proposer un ordonnanceur, capable de répartir les missions entre les robots. Cet ordonnanceur devra pouvoir s'adapter de façon dynamique aux changements qui peuvent survenir. Le principal changement est l'arrivée de nouvelles missions à affecter, envoyées par le WMS. Il peut aussi s'agir de s'adapter aux perturbations décrites dans la section 1.2.6, qui auront un impact sur la disponibilité des robots. Ainsi, les affectations devront être faites en ligne, c'est à dire recalculées durant le cours de l'exécution des missions.

Afin de comparer les performances, mais aussi de contrôler le respect des différentes contraintes, il est aussi nécessaire de disposer d'un simulateur apte à jouer les différents scénarios afin de calculer des indicateurs. Le projet est donc composé tel que décrit par la Figure 3.1.

Les différentes missions sont transmises par le WMS. L'ordonnanceur, les recevant, détermine l'ensemble des affectations. Ces affectations sont ensuite simulées tout en sauvegardant des informations à propos des robots (position, mission en cours, niveau de batterie, ...) au cours de cette simulation. Cet historique est ensuite exploité afin de fournir des indicateurs permettent de visualiser le déroulement des missions et de contrôler aussi bien les performances que le respect de la globalité des contraintes.

3.1 Ordonnanceur

Un ordonnancement réalisable doit respecter les conditions suivantes :

1. chaque mission est réalisée exactement une fois par un seul véhicule,
2. chaque itinéraire est faisable sur le plan énergétique, c'est-à-dire que le niveau de batterie d'un robot lorsqu'il entre et sort de n'importe quel endroit se situe entre 0% et 100%,

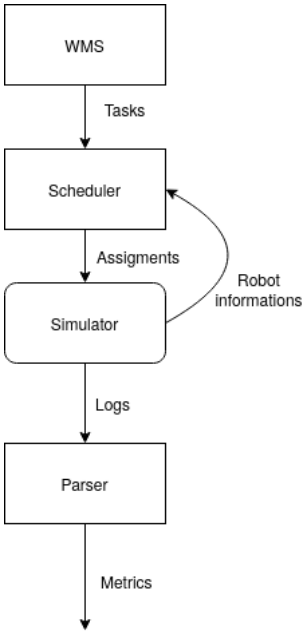

FIGURE 3.1 – Schéma composants du projet

3. Deux robots ne se chargent pas simultanément à la station de recharge.

L'objectif est de réduire au minimum le temps total nécessaire pour servir tous les clients, c'est-à-dire, le makespan.

Définition 3.1 (Makespan)
En recherche opérationnelle, le makespan est la durée totale d'un projet. C'est donc la distance dans le temps qui s'écoule du début à la fin du travail.

On considère qu'il existe deux types de missions, les missions de recharge de batterie et les missions de transport. Afin de simplifier le problème, et donc diminuer les temps de calcul, ces deux types de missions seront gérés indépendamment.

On s'intéressera dans un premier temps à la gestion des batteries. On s'emploiera ensuite à l'affectation des missions de transport aux robots disponibles, c'est-à-dire ayant un niveau de batterie suffisant.

3.1.1 Gestion des batteries

On définit un seuil de batterie threshold, dépendant de la topologie de l'entrepôt, en dessous duquel un robot ne pourra pas se voir affecter une mission de transport. Ce seuil

correspond à trois fois l'énergie nécessaire pour relier les deux points les plus éloignés de l'entrepôt. Ainsi, si un robot se voit affecter une mission, il pourra, au minimum, l'exécuter et rejoindre la station de recharge sans risquer de tomber en panne de batterie, et ce, quelle que soit la mission.

La station de recharge de batterie est une ressource disjonctive et qui n'est présente qu'en un seul exemplaire dans un entrepôt.

Définition 3.2 (Ressources disjonctives)
Ressources qui ne peuvent exécuter qu'une tâche à la fois.

Il est donc indispensable de planifier les missions de recharge de batterie des robots de façon à ce que deux robots ne se voient pas affecter une telle mission simultanément. Pour cela, quand la station de recharge de batterie est disponible, on sélectionne un robot auquel on affecte la mission de recharge de batterie.

Pour ce faire, on aura préalablement déterminé le niveau de batterie, ceiling (seuil maximum), en dessous duquel un robot pourra se voir affecter à une mission de recharge de batterie. On sélectionnera alors le robot ayant le niveau de batterie le plus faible et en dessous de ceiling. Le robot sélectionné *(s'il existe)* se voit alors affecter une mission de recharge de batterie. Ce processus est expliquer à l'aide de la Figure 3.2.

Remarque 3.1
Si tous les robots ont un niveau de batterie élevé (supérieur à ceiling), aucun robot n'est sélectionné.

On propose ensuite deux manières différentes de gérer les missions de recharge de batterie :

1. une mission de recharge de batterie peut être interrompue, dans ce cas, on considère qu'un robot reste disponible durant une mission de recharge. Il pourra donc se voir attribuer une mission de transport, ce qui écrasera la mission de recharge de batterie à n'importe quel moment (avant ou pendant le chargement).
2. une mission de recharge de batterie ne peut pas être interrompue, dans ce cas, dès lors qu'un robot est sélectionné pour une mission de recharge de batterie, il sera indisponible pour toute autre mission, jusqu'à la charge complète de sa batterie.

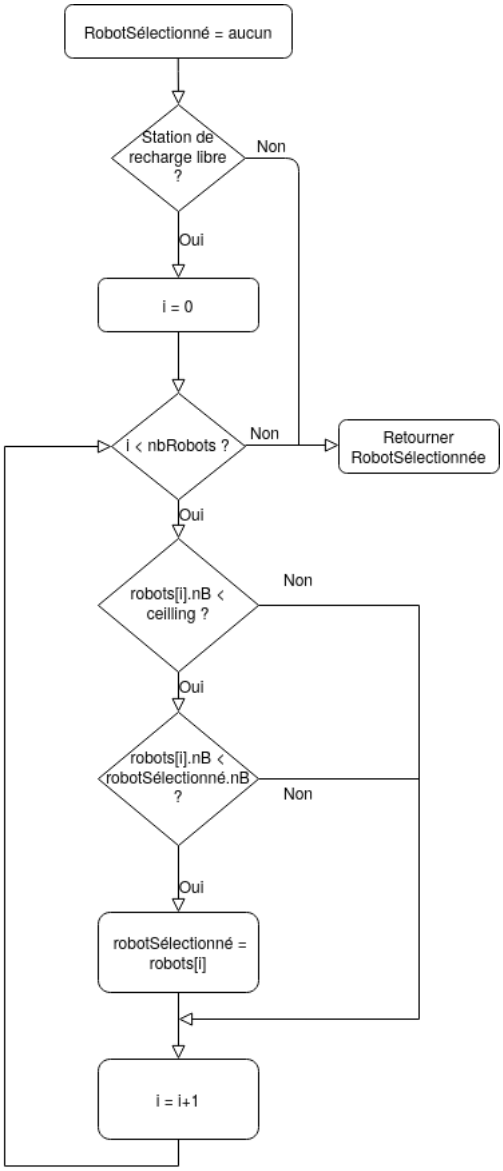

FIGURE 3.2 – Logigramme gestion de l'accès à la station de recharges des batteries

3.1.2 Affectation des missions de transport

Les affectations des missions de transport sont, elles aussi, réalisées en ligne. Des calculs sont effectués si :

— un événement survient (fin d'une mission dans le simulateur, arrivée d'une nouvelle mission depuis le WMS),

— aucun calcul n'a été effectué depuis X unité de temps.

Les affectations sont fixées Y unité de temps avant la date de début prévue, c'est à dire qu'elles ne sont pas recalculées si leur date de début est trop proche. A chaque calcul, seules les affectations des N prochaines missions sont calculées.

Représentation du problème par un graphe

Notre problème est un problème de pick-up and delivery, cependant il ne correspond pas totalement au modèle général décrite dans la section 2.2. En effet, dans notre problème :

- les robots ne sont pas stockés dans un dépôt (qui correspondrait à leur position initiale),
- lorsqu'un robot est affecté à une mission, il réalise obligatoirement le déplacement du point de prise au point de pose de cette mission.

Ainsi, il est possible de modéliser ce problème différemment du problème général de pick-up and delivery.

On formule le problème à l'aide d'un nouveau graphe $G = (V, A)$. L'ensemble des nœuds V est partitionné en $\{K, M\}$ tel que $K = \{R_1, \ldots, R_n\}$ représente les positions initiales des n robots et $M = \{M_1, \ldots, M_N\}$ représente les N missions. Une mission M_α est un couple (P_α, D_α), avec P_α son point de prise et D_α son point de pose. Un coût m_α est associé à chaque mission, il représente le coût du trajet entre le point de prise et le point de pose de la mission α.
Un arc (i, j) est dans le graphe G si l'une des conditions suivantes est satisfaite :

- $i \in K, j \in M$,
- $i \in M, j \in M$.

Le coût pour traverser l'arc (i, j) est noté c_{ij}, il représente le coût du trajet entre le point de pose de la mission i, D_i, et le point de prise de la mission j, P_j.

Par exemple, si on considère qu'on dispose de 2 robots pour 4 missions, on obtient le graphe illustré par la figure 3.3.

L'objectif est de trouver pour chaque robots un parcours dans ce graphe de sorte que, chaque nœud soit visité exactement une fois (comme illustré par la Figure3.4).

Dans la solution de la figure 3.4, le robot R_1 va d'abord réaliser la mission M_1 puis la mission M_4 alors que le robot R_2 va réaliser la mission M_3 puis la mission M_2.

Il existe plusieurs solutions à ce problème d'affectation, le but dans notre cas va être de trouver celle qui minimisera la durée totale d'exécution des missions, le makespan.

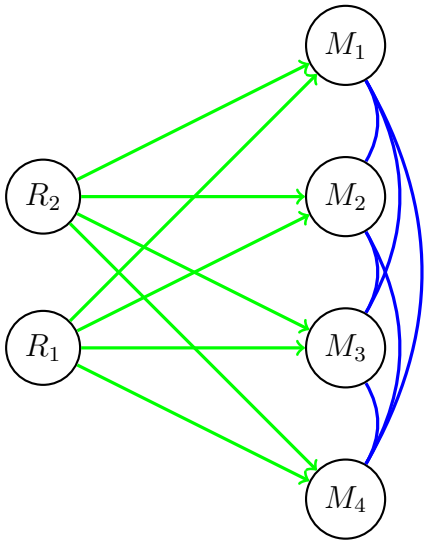

FIGURE 3.3 – Exemple de modélisation 2 robots et 4 missions.

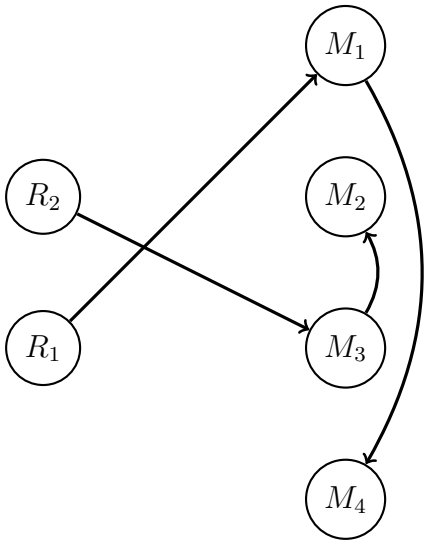

FIGURE 3.4 – Exemple de solution 2 robots et 4 missions.

Deux méthodes d'affectation ont été développées. Une première méthode utilise la programmation linéaire en nombres entiers (PLNE) pour l'affectation des lots de N missions de manière optimale. Une seconde méthode détermine mission par mission quel robot devra l'effectuer.

3.1.3 Méthode 1 : Programmation linéaire en nombres entiers

En considèrent la modélisation exposée dans la section 3.1.2, on peut réduire le problème à un problème de programmation linéaire en nombres entiers. Cette formulation est inspirée de celle réalisée pas Cordeau [2].

Dans ce modèle, on utilise une variable de décision binaire x_{ij}^k égale à 1 si et seulement si l'arc (i, j) est traversé par le robot $k \in K$.

Remarque 3.2
Avec cette hypothèse, on peut en déduire les assertions suivantes :

1. *La proposition $x_{i,j}^1 = 1$ signifie que l'arc qui lie les missions M_i et M_j est emprunté par le robot R_1 au cours de sont trajet.*
2. *La proposition $\sum_{k \in K} x_{i,j}^k = 1$ signifie qu'exactement l'un des robots de K emprunte l'arc qui lie les missions M_j et M_j.*

$$Minimize \quad makespan \tag{3.1}$$

tel que

$$\forall j \in M \quad \sum_{k \in K} \sum_{i \in V} x_{ij}^k = 1 \tag{3.2}$$

Soit M un entier "grand"

$$\forall k \in K \quad \sum_{j \in M} x_{kj}^k \times M > \sum_{j \in V} \sum_{i \in V} x_{ij}^k \quad et \quad \sum_{i \in K} \sum_{j \in V} x_{ij}^k \leqslant 1 \tag{3.3}$$

$$\forall i \in V \quad \sum_{j \in V} \sum_{k \in K} x_{ij}^k \leqslant 1 \tag{3.4}$$

$$\forall i \in M \quad \forall k \in K \quad \sum_{j \in V} x_{ij}^k \leqslant \sum_{j \in V} x_{ji}^k \tag{3.5}$$

$$\forall k \in K \quad makespan \geq \sum_{i,j \in V} (c_{ij} + m_j) x_{ij}^k \tag{3.6}$$

La contrainte 3.2 permet de garantir que chaque mission est visitée une et une seule fois, par l'un des robots. Soit celui-ci arrive de l'un des points initiaux ($i \in K$), dans ce cas, il emprunte un arc vert de la Figure 1, soit il arrive de l'une des autres missions ($i \in M$), dans ce cas, il emprunte un arc bleu de la Figure 1.

Remarque 3.3
Soit $k \in K$ un robot.
$\sum_{j \in V} \sum_{i \in V} x_{ij}^k > 0 \iff$ *le robot k emprunte au moins un arc.*
$\sum_{j \in M} x_{kj}^k > 0 \iff$ *le robot k emprunte au moins un arc de sa position initiale vers une mission.*
$\sum_{i \in K} \sum_{j \in V} x_{ij}^k \leqslant 1 \iff$ *le robot k emprunte maximum un arc de sa position initiale vers l'une des missions.*

La contrainte 3.3 permet de garantir que si le robot k est utilisé, alors il débutera sont trajet de sa position initiale.

La contrainte 3.4 permet de garantir qu'un nœud sera au maximum une fois l'origine d'un arc emprunté.

La contrainte 3.5 permet de garantir qu'aucun robot k ne pourra quitter un point de prise ou de pose s'il ne l'a pas atteint.

La contrainte 3.6 permet de définir le makespan comme étant le plus grand des temps de trajet pour réaliser l'ensemble des missions d'un robot.

L'objectif, décrit par 3.2 est alors de minimiser ce plus grand temps de trajet.

Élimination des sous-tours

Les solution générée par la modélisation du PLNE alors proposée peuvent contenir des "sous tours". Ces "sous-tours" rendent impossible l'enchaînement des missions(voir Figure 3.5)

Sur la Figure 3.5, on peut voir un sous-tour entre les missions 2, 3 et 4 et il n'y a pas d'arc qui les lie à la mission 1. Cette solution du PLNE n'en est pas une pour le problème d'affectation.
Il faut alors introduire des contraintes d'élimination des sous-tours, tel que la contrainte 3.7

$$\forall S \subset M \quad tq \quad \leq |S| \leq n-2 \quad \forall k \in K \quad \sum_{(i,j) \in S} x_{ji}^k \leq |S| - 1 \tag{3.7}$$

Or cela induit d'introduire un nombre de contraintes qui croit de façon exponentielle avec le nombre de nœuds. Ce qui limite son utilisation à des instances de petite taille.

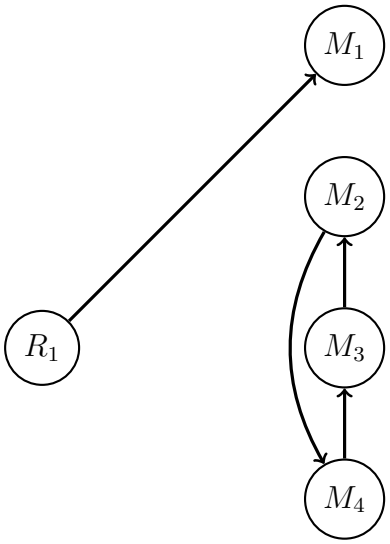

FIGURE 3.5 – Exemple de problème de sous-tour pour 1 robots et 4 missions.

Pour corriger cela, il existe au moins deux solutions.

1. Formulation compacte MTZ

   L'une d'elles utilise la formulation compacte MTZ (Miller, Tucker, Zemlin). Cette technique consiste à ajouter une variable U_i avec $i \in V$, entière, pour chaque nœud du graphe. Afin d'éviter les boucles, la suite des valeurs U_i le long du parcours des robots sera croissante. Ceci se traduit par la contrainte suivante :
   Soit M un entier suffisamment grand,

$$\forall j \in V \quad \forall i \in V \quad U_i - U_j + 1 \leqslant M(1 - \sum_{k \in K} x_{ij}^k) \tag{3.8}$$

   Remarque 3.4

   (a) $\sum_{k \in K} x_{ij}^k = 1$, *signifie que l'arc qui lie le point i et le point j est emprunté par exactement un robot au cours de son trajet et donc que le point i est visité avant le point j.*

   (b) si la contrainte (3.8) est vraie, on à les équivalences suivantes : $\sum_{k \in K} x_{ij}^k = 1 \iff U_j \geq U_i + 1 \iff U_j > U_i$.

   Un nœud j visité avant un autre nœud i, aura une variable U_j associée strictement inférieure à celle du nœud i, $U_j < U_i$.
   Il ne peut donc pas y avoir de boucle dans le chemin des robots (aucun arc ne pourra jamais lié un nœud i à un autre nœud j déjà visité, car cela contredirait la contrainte $U_j > U_i$ induite par (3.8)).

Exemple 3.1
Le sous tour $x_{24} = 1, x_{43} = 1, x_{32} = 1$ *est éliminé car sinon* $U_2 < U_4 < U_3 < U_2$

2. Lazy constaintes

   Une autre solution consiste à utiliser les contraintes 3.7 mais en les définissant comme des "contraintes paresseuses". Ainsi, elles seront ignorées lors de la résolution du problème par le solver. Elles restent inactives jusqu'à ce qu'une solution réalisable soit trouvée, auquel moment on teste si la solution trouvée vérifie ces contraintes. Si la solution viole une des contraintes paresseuses, la solution est rejetée et cette contrainte paresseuse violée est intégrée dans le modèle actif.

3.1.4 Méthode 2 : Ordonnancement mission par mission

Dans cette section, les missions sont traitées une à une dans l'ordre dans lequel elles sont fournies par le WMS. Pour une mission donnée, on va calculer pour chaque robot un score en fonction de son niveau de batterie, de sa position et sa date de disponibilité.

Pour cela, on normalise le niveau de batterie entre deux bornes (une borne inférieure : threshold et une borne supérieure : ceiling) à l'aide d'une fonction batteryNorme().

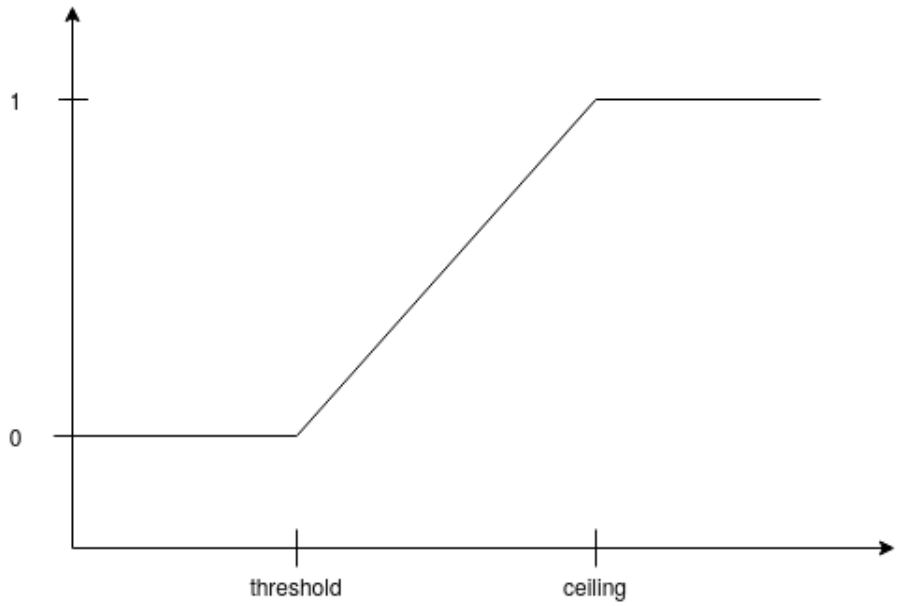

FIGURE 3.6 – Représentation de la fonction de normalisation du niveau de batterie

On estime ensuite le temps de trajet entre la position finale du robot et la position du début de la mission, le point de prise. On ajoute ce temps à la date de disponibilité de ce robot pour connaitre la date à laquelle il pourra débuter la mission. On note cette date availabilityDate .

Le score est ensuite calculé en faisant le quotient du niveau de batterie normalisé par la date de début de mission : score = batteryNorme/availabilityDate.

Le robot ayant le plus haut score se voit alors affecté la mission. Ceci est répété pour les N missions qui doivent être affectées.

3.2 Simulateur et Indicateurs

3.2.1 Simulateur

Le simulateur simule l'exécution des différentes missions tout en faisant varier le niveau de batterie et la position des robots ainsi qu'en simulant le temps qui passe. Toutes les informations sont enregistrées dans un fichier dont l'analyse permet la mise en place d'indicateurs.

3.2.2 Indicateurs

Dans cette sous-section, on présente les différents indicateurs qui vont être calculés : un graphique permettant de mettre en évidence l'évolution du niveau des batteries et un diagramme donnant la possibilité de visualiser dans le temps les diverses missions.

Évolution du niveau des batteries

Contrôler l'évolution du niveau des batteries est important car il faut s'assurer que la solution simulée est faisable sur le plan énergétique, c'est-à-dire que le niveau de batterie d'un robot se situe toujours entre 0% et 100%.

Dans la Figure 3.1.1 chaque courbe de couleur représente l'évolution de la batterie de l'un des robots au cours du temps.

Exécution des missions

Un autre diagramme permet de visualiser dans le temps les diverses missions réalisées par chaque chariot.

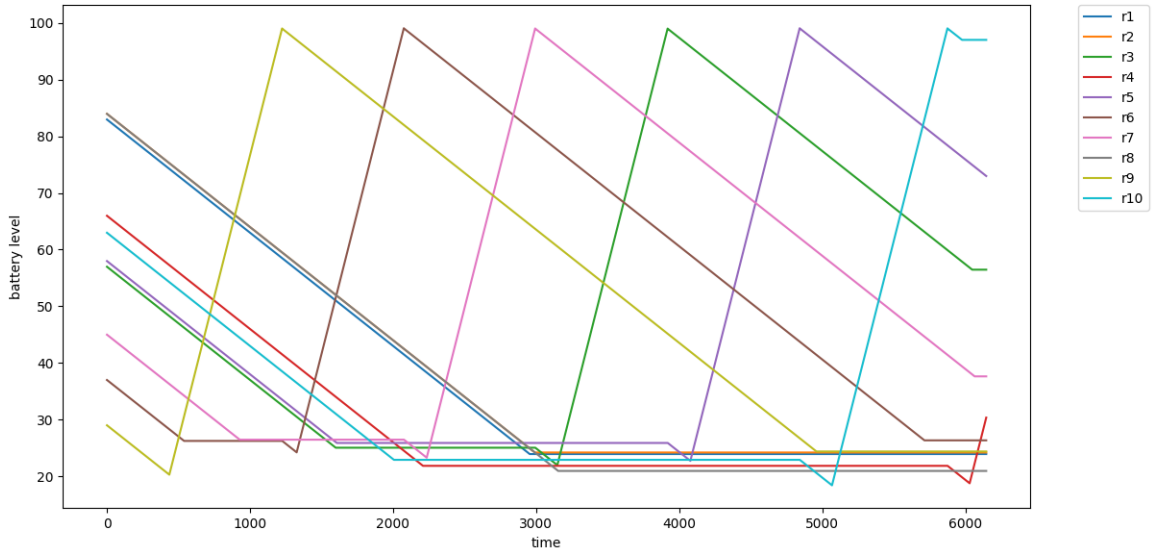

FIGURE 3.7 – Exemple d'indicateur représentant l'évolution des niveaux de batterie.

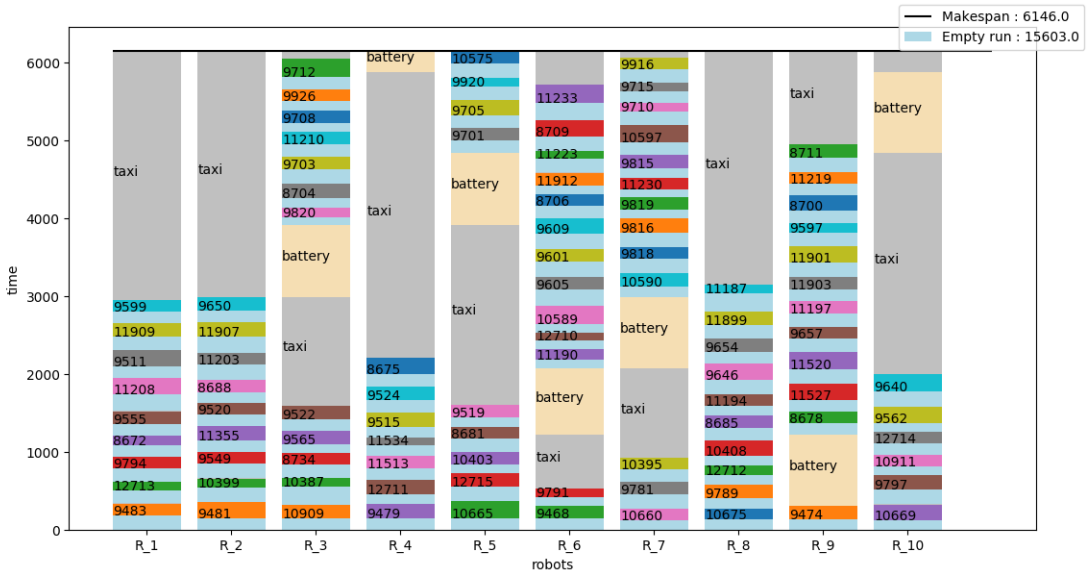

FIGURE 3.8 – Exemple d'indicateur représentant l'exécution des mission

Dans la Figure 3.8, on trouve en abscisses les robots et le temps en ordonnées. Les cases

colorées correspondent aux missions (avec leur numéro d'identification). Les zones bleues claires représentent les temps de parcours pour atteindre ces missions, les zones beiges les temps de recharge de batterie et les zones grises sont les temps où les chariots sont inactifs. Ce diagramme permet d'afficher les performances avec le makespan et les temps de trajet à vide cumulés (en haut à droite). Mais on peut aussi visualiser le respect des contraintes, comme le fait de ne pas avoir deux chariots en charge simultanément.

3.2.3 Conclusion

Dans ce chapitre, on a expliqué la façon dont on gère les niveaux de batterie et on a proposé deux façons d'affecter les missions de transport aux robots. On a aussi présenté deux indicateurs permettant de contrôler les affectations. Dans le chapitre suivant, on se propose de comparer les deux techniques d'affectation de missions.

Chapitre 4

Résultats

L'objectif de nos expériences de calcul est d'évaluer et de comparer les performances des deux ordonnanceurs présentés afin de déterminer dans quels cas il est préférable d'utiliser l'un plutôt que l'autre pour les installations robotisées.

4.1 Conditions des tests

Pour les tests, on s'appuie sur un ensemble de 4200 missions dans un entrepôt disposant de 10 robots. On a considéré un banc d'essai de 80 instances. Ce banc d'essai comprend quatre séries de 20 instances, chacune ayant 10, 20, 50 ou 100 missions.

Dans cet entrepôt, le seuil de batterie en dessous duquel un robot ne pourra pas se voir affecter une mission de transport (threshold, voir Section 3.1.1) est fixé à 30%. Et le niveau de batterie, en dessous duquel un robot pourra se voir affecter à une mission de recharge de batterie (ceiling) est fixé à 50%.

En ce qui concerne la méthode utilisant la programmation linéaire en nombres entiers, le nombre maximal de missions pouvant être affectées par une itération correspond au nombre total de robots, soit 10 missions.

4.2 Missions connues dès le début des calculs

Dans cette section, on considère que l'ensemble des missions à affecter sont connues dès le début des calculs. Le Tableau 4.1 présente les résultats. La colonne *time* représente le temps nécessaire pour réaliser l'ensemble de la simulation, c'est-à-dire les temps de calculs cumulés. La colonne *makespan* représente la durée totale nécessaire pour réaliser

les missions. Enfin, la colonne *empty run* représente le temps de trajet à vide des robots entre deux missions. Pour tous ces temps, on donne la moyenne μ ainsi que l'écart-type σ, pour les 20 instances, exprimés en secondes.

|I|	Score						PLNE					
	makespan		empty run		time		makespan		empty run		time	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
10	568	38	2430	235	13	18	529	37	2330	140	20	24
20	935	42	4237	302	28	41	902	32	4148	294	37	44
50	2162	133	8114	1020	75	68	1980	64	9392	949	171	180
100	6143	195	19894	2939	70	66	5971	126	21079	3608	488	297

TABLE 4.1 – Résultats des calculs pour les ordonnanceurs sur les instances de 10, 20, 50 et 100 missions, connues dès le début des calculs

À la lecture du Tableau 4.1, on constate, que pour des instances du problème de petite taille, 10 ou 20 missions, les temps de calcul (*time*) sont comparables pour les deux méthodes. Cependant, pour des instances de grande taille, 50 ou 100 missions, les temps de calcul pour la méthode utilisant la programmation linéaire en nombre entier explosent comparativement à celle utilisant le score. Ce qui peut rendre inexploitable la méthode $PLNE$ pour les installations dont le nombre de missions est important et ayant une capacité de calculs limitée. Ces résultats nous permettent aussi de constater que, pour une méthode donnée, les temps de calculs sont très variables. En effet les écarts-types sont très grands.

Au prix de calculs plus longs, la méthode du $PLNE$ obtient de meilleures performances en terme de makespan, dans 91% des cas. Concernant, le temps de trajet à vide (*empty run*) il est plus faible avec la méthode du $PLNE$ dans seulement 52% des tests effectuer pour ce banc d'essai.

4.3 Missions connues une à une

Dans cette section, on considère que les missions à affecter sont connues une à une. Toutes les 50 secondes, une nouvelle mission est fournie par le WMS. Le Tableau 4.2 présente les résultats.

Le Tableau 4.2 nous permet de constater que, dans le cas où l'on connait les missions une à une, la méthode utilisant la programmation linéaire en nombres entiers est efficace seulement lorsque le nombre de missions est élevé. En effet, cette méthode présente un intérêt seulement si plusieurs missions doivent être affectées. Cette situation se produit quand il y a de nombreuses missions, car une accumulation se produit.

|I|	Score				PLNE			
	makespan		empty run		makespan		empty run	
	μ	σ	μ	σ	μ	σ	μ	σ
10	788	34	2346	473	839	42	2568	671
20	1318	53	5146	732	1349	66	5135	966
50	3104	124	11677	912	2870	57	11163	778
100	6791	180	20851	3053	6507	193	21869	3703

TABLE 4.2 – Résultats des calculs pour les ordonnanceurs sur les instances de 10, 20, 50 et 100 missions, connues une à une

Ces expériences ont aussi permis d'étudier les temps de calcul moyen nécessaires aux deux méthodes, ceci est représenté par la Figure 4.1. On constate alors que, contrairement à la méthode du score, le nombre d'affectations influence fortement les temps de calcul pour la méthode utilisant la programmation linéaire en nombres entiers.

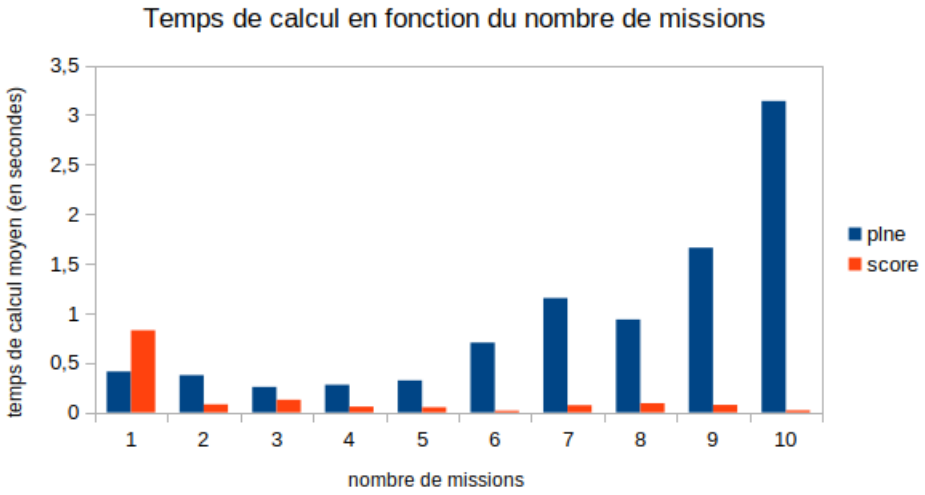

FIGURE 4.1 – Temps de calcul en fonction du nombre de missions

4.4 Conclusion

Les tests effectués ont permis de mettre en évidence le fait que la méthode de programmation linéaire en nombres entiers ne présente un intérêt, sur la méthode du score, que lorsque les affectations concernent plusieurs missions. Cependant, lorsque le nombre de missions augmente, les temps de calculs deviennent problématiques pour cette méthode.

Conclusion et perspectives

Ce rapport donne un aperçu du travail réalisé dans le cadre du master SDS en partenariat avec l'entreprise Balyo. L'objectif est de proposer une solution permettant d'affecter les missions de transport à des chariots de manutention robotisés.

Nous nous sommes intéressés, dans un premier temps, à formaliser le problème et mener une étude bibliographique. Ceci nous a permis de déterminer les solutions existantes et les termes utilisés dans la communauté scientifique pour des problèmes comparables. Ainsi, nous avons pu rapprocher le problème d'affectation des tâches à un *Pickup and Delivery Problem*. Toutes ces recherches ont ensuite servi de support dans l'implémentation de solutions. En effet, dans un second temps, nous avons développé un procédé pour la gestion de la batterie des chariots ainsi que deux méthodes d'ordonnancement. Ces ordonnancements ont ensuite été simulés afin d'en comparer les performances.

Afin d'évaluer les méthodes développées dans ce projet, il serait intéressant de les tester sur des problèmes de référence (benchmarks), telles que les instances de Montoya [10].

Les mois qui vont suivre vont permettre d'approfondir le travail réalisé et de proposer des solutions pour des problèmes avec des contraintes qui n'ont pas encore été prises en compte :

— robots de différents types ayant leurs propres caractéristiques,

— mission de différentes priorités.

A terme ce travail permettra à l'entreprise d'intégrer un ordonnanceur performant et adapté aux installations qu'ils proposent.

Bibliographie

[1] A green vehicle routing problem. *Transportation Research Part E : Logistics and Transportation Review*, 48(1) :100 – 114, 2012. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[2] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54 :573–586, 06 2006.

[3] Jacques Desrosiers, Yvan Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6, 02 1986.

[4] Aurelien Froger, Jorge E Mendoza, Ola Jabali, and Gilbert Laporte. The electric vehicle routing problem with capacitated charging stations. 2019.

[5] Shaza Hanif, Rinde Lon, Ning Gui, and Tom Holvoet. Delegate mas for large scale and dynamic pdp : A case study. volume 382, pages 23–33, 01 2011.

[6] Imen Harbaoui Dridi, Ryan Kammarti, Mekki Ksouri, and Pierre Borne. Approche multicritère pour le problème de ramassage et de livraison avec fenêtres de temps à plusieurs véhicules. 03 2009.

[7] Zoulel Kouki, Besma Fayech, and Mekki Ksouri. Extended cnp framework for the dynamic pickup and delivery problem solving. volume 296, pages 61–71, 07 2009.

[8] Jan Lenstra and A. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11 :221 – 227, 10 2006.

[9] Haibing Li and Andrew Lim. A metaheuristic for the pickup and delivery problem with time windows. volume 12, pages 160–167, 12 2001.

[10] Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B : Methodological*, 103 :87 – 110, 2017. Green Urban Transportation.

[11] William Nanry and John Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B : Methodological*, 34 :107–121, 02 2000.

[12] Harilaos Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14 :130–154, 05 1980.

[13] Harilaos Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17 :351–357, 08 1983.

[14] Mais Rachid, Wahiba Ramdane Cherif, Christelle Bloch, and Pascal Chatonnay. Classification de problèmes de tournées de véhicules. pages 1149–1158, 03 2008.

[15] Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49 :258–272, 07 2007.

[16] Martin Savelsbergh and Marc Sol. Drive : Dynamic routing of independent vehicles. *Operations Research*, 46, 02 1997.

[17] Thomas Sexton and Young-Myung Choi. Pickup and delivery of partial loads with "soft" time windows. *American Journal of Mathematical and Management Sciences*, 6, 02 1986.

Résumé — Balyo est une entreprise qui travaille dans le domaine de l'automatisation des chariots de manutention. L'objectif du projet est de proposer une solution pour optimiser le flux des chariots dans les entrepôts. Concrètement, cela signifie attribuer des tâches aux chariots de façon à réduire le temps nécessaire à leur réalisation. Dans un premier temps, une description détaillée du projet et une présentation d'une installation type sont proposées. Ensuite, une étude bibliographique de l'état de l'art est réalisée. Elle aborde les questions de l'attribution des tâches de ramassage et de livraison et le problème de tournées de véhicules électriques. Deux méthodes pour l'affectation de missions sont alors proposées ainsi que des indicateurs permettant de contrôler les solutions ainsi obtenues.

Mots clés : entrepôt automatisé ; affectation des tâches ; tâche de ramassage et de livraison ; problème de tournées de véhicules électriques.

Abstract — Balyo is a company working in the field of automation of industrial trucks for the transport of pallets.The aim of the project is to propose a solution to optimize the flow of trucks in warehouses. In concrete terms, this means assigning tasks to trucks in order to reduce the time required to complete them. As a first step, a detailed description of the project and a presentation of a typical installation are proposed. Then a bibliographical study of the state of the art is carried out. It deals with the questions of pickup and delivery task assignment and electric vehicle routing problem. Two methods for mission assignment are then proposed as well as indicators to monitor the resulting solutions.

KeyWords : automated warehouse ; task assignment ; pickup and delivery task ; electric vehicle routing problem.

Polytech Angers
62, avenue Notre Dame du Lac
49000 Angers