Optimization of a fleet of reconfigurable robots for logistics warehouses

Mari Chaikovskaia




Université Clermont Auvergne

École doctorale Sciences Pour l'Ingénieur

Spécialité "Informatique"

Thèse de doctorat

Optimization of a fleet of reconfigurable robots for logistics warehouses

Présentée par

Mme Mari Chaikovskaia

Soutenance prévue le 25 septembre 2023 devant le jury composé de :

Pierre DAVID	Professeur des universités, Grenoble INP	Rapporteur
Khaled HADJ-HAMOU	Professeur des universités, INSA Lyon	Rapporteur
Marie-Pierre GLEIZES	Professeur des universités, Université Toulouse III	Examinatrice
Jean-Philippe GAYON	Professeur des universités, Université Clermont Auvergne	Directeur de thèse
Jean-Christophe FAUROUX	CEO MecaBotiX / Chercheur associé, Institut Pascal	Co-encadrant
Zine Elabidine CHEBAB	CTO MecaBotiX / Chercheur associé, Institut Pascal	Co-encadrant

INP Clermont Auvergne

1 Rue de la Chebarde, 63178 Aubière, France

Acknowledgements

This work was financed by the Auvergne Rhône Alpes Regional Council as part of a Pack Ambition Research on the RoMoCo project (Cooperative Modular Robots).

Remerciements

En 2019, je suis venue en M2 de Genie Industrielle sans visa d'étudiant et je pouvais à peine dire «Je m'appelle Mari». Merci beaucoup à *Sylvie Norre* et *Olivier Devise* de m'avoir accepté sur cette formation et de m'avoir littéralement donné une nouvelle vie.

Merci à l'équipe de mes encadrants, *Jean Philippe Gayon*, *Jean-Christophe Fauroux*, *Zine Elabidine Chebab*. J'ai beaucoup appris de vous. Pendant ces 3 ans, j'ai senti la liberté et la responsabilité, et c'est ce qui m'a beaucoup motivée. Au cours de ces trois années, il y a eu parfois des moments difficiles de la vie, mais j'ai toujours ressenti votre compréhension et votre soutien.

Je tiens à remercier le directeur de ma thèse, *Jean Philippe Gayon*, d'avoir cru en moi, même si mes études à la base étaient très éloignées du sujet de la thèse. C'est une grande chance d'écrire une thèse sous votre direction. Merci d'avoir toujours écouté mes idées, permis de ne pas être d'accord avec les vôtres, de discuter, d'apprendre de mes erreurs. Merci pour l'ambiance très conviviale. Merci pour toutes les possibilités et toute l'aide précieuse que vous m'avez donnée.

Un grand merci *Alain Quilliot* pour votre ouverture, votre gentillesse, votre aide et vos explications. Merci à l'équipe du *LIMOS* qui est devenue une famille pour moi. Votre soutien, votre sourire m'ont aidé à me sentir à la maison.

Merci beaucoup à mes *amis*. Milles mercis aux gens qui soutiennent toujours toutes mes initiatives courageuses, me donnent infiniment d'amour et de soutien, et grâce auxquels j'avance : ma *Mère*, mon *Père* et mon *Conjoint*.

J'ai une grande chance de la vie d'être toujours entourée par les gens très gentils et bienveillants. Merci pour les meilleures années que j'ai vécue.

Abstract

This thesis is interested in the robotization of operations in logistics warehouses, focusing on transport operations from reception area to storage area and vice versa. The transport operations are performed by reconfigurable poly-robots which consist of elementary robots that can be assembled in different ways over time to adapt to the loads to be transported. Each poly-robot configuration has its own transportation capacity. After each transportation, poly-robots can be reconfigured to adapt to the load type. We first consider the fleet sizing problem which consists in determining the optimal number of elementary robots required to transport a set of loads within a specified time frame. We formulate several variants of this problem as integer linear programs. We study the computational complexity of this problem and provide a polynomial heuristic algorithm. We show how reconfigurability can allow to diminish the number of required elementary robots. Finally, we also study the problem of scheduling such a fleet with the objective to minimize the time to execute all transportation tasks.

Résumé en français

Optimisation d'une flotte de robots reconfigurables dans un entrepôt logistique

La thèse s'intéresse à un système industriel comportant plusieurs entités robotiques de différentes natures. On considère plus particulièrement la robotisation des opérations dans les entrepôts logistiques, en se focalisant sur les opérations de transport (transfert de la zone de réception vers la zone de stockage et vice-versa).

Robotisation des entrepôts

Il existe de nombreux entrepôts où des tâches laborieuses et répétitives sont effectuées manuellement : chargement et déchargement des camions, contrôle, déplacement des charges, préparation des commandes, emballage, etc. Il est prévu que 1,5 million d'emplois soient remplacés par des robots entre 2016 et 2026 dans la zone euro et que les coûts de manutention soient réduits de 20 à 40% [Roland Berger, 2016]. Les raisons principales de la robotisation des entrepôts sont : la difficulté de recruter des personnels pour le transport de charges, notamment des opérateurs caristes détenant le certificat d'aptitude à la conduite en sécurité [INRS, 2023]; le faible taux de fidélité des opérateurs contractuels sur des métiers de manutention jugés pénibles et peu motivants; le besoin d'augmenter la productivité des entrepôts, dans un contexte économique où le coût du m² d'entrepôt augmente chaque année fortement. De plus, étant donné la réduction des coûts de mise en place et d'entretien des robots, il n'est pas surprenant que les entrepôts logistiques soient de plus en plus robotisés. Ainsi, l'utilisation de robots dans les en-

trepôts devient courante et nous nous posons la question de l'intérêt de la coopération des robots.

Coopération et reconfiguration

Les avantages de la coopération sont démontrés dans le monde animal [Dugatkin, 1997]. La coopération peut être de différents types : l'exécution d'une tâche par plusieurs subordonnés sous la direction d'un leader ou l'exécution d'une même tâche avec le même niveau de responsabilité [Theraulaz et al., 2002; Fourcassié et al., 2010]. L'approche coopérative peut être appliquée non seulement dans la nature, mais dans l'industrie [Le et al., 2010]. Des robots coopératifs capables de travailler en parallèle sur une même tâche ouvrent de larges perspectives [Noreils, 1992]. Dans l'industrie, la coopération peut être définie comme l'exécution conjointe d'une tâche [Jung et al., 1998]. Tuci et al. [2018] constatent qu'il y a coopération si la tâche ne peut être exécutée séquentiellement par un seul robot et nécessite une coordination des actions et une communication entre robots. Dans cette thèse, nous utilisons cette dernière définition.

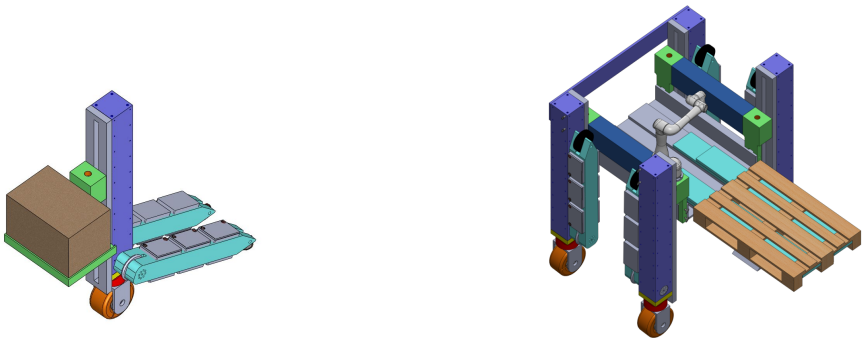
Le second aspect de cette étude est la reconfigurabilité des robots. On définit la reconfigurabilité comme l'aptitude de robots élémentaires à être attachés ou détachés les uns les autres, par un opérateur ou de façon autonome, pour modifier la configuration du robot résultant. La principale différence avec les robots coopératifs est que les robots coopératifs restent dans leur configuration tout au long de l'horizon du temps, tandis que la configuration des robots reconfigurables peut être modifiée au cours du temps.

Les robots reconfigurables et coopératifs offrent des solutions prometteuses pour répondre aux demandes en optimisant le transport des charges au sein d'un entrepôt.

Collaboration industrielle

Ce travail, inspiré par la thèse de Chebab [2018], est le fruit d'une collaboration de recherche avec la startup MecaBotiX. Chebab [2018] a travaillé sur la conception de nouvelles structures de robots mobiles modulaires qui coopèrent afin de réaliser des tâches liées à la gestion et au transport de caisses. La société MecaBotiX s'occupe de la conception et la vente de robots, notamment des robots M3-Cooper qui sont des Robots **M**anipulateurs **M**obiles **M**odulaires **Coopér**atifs. Ils sont capables de s'agréger en grappe pour constituer un poly-robot adapté à la charge à déplacer. Ils peuvent

s'adapter au type de produit (taille/masse) et naviguer en autonomie dans des environnements tels que des entrepôts, des sites de production ou des chantiers. La figure 1 montre deux exemples de poly-robots M3-Cooper transportant des charges. Le mono-bot de la figure 1a peut transporter une caisse tandis que le quadri-bot de la figure 1b peut transporter une palette. Les robots permettent de décharger les opérateurs des tâches pénibles à faible valeur ajoutée et de les re-déployer sur les tâches complexes : contrôle qualité, préparation de commande. Ils permettent aussi des créations d'emploi dans la supervision et la maintenance robotique.



(a) Mono-bot (b) Quadri-bot

Figure 1: Exemples de poly-robots reconfigurables M3-Cooper [MecaBotiX, 2023]

Problématique

Dans ce travail, nous examinons le transport des charges par des robots reconfigurables, nommés poly-robots, constitués de robots élémentaires. Un robot élémentaire ne peut pas être divisé en plusieurs robots. Un poly-robot est un groupe de robots élémentaires qui sont agrégés afin d'effectuer conjointement une tâche comme un seul robot. Après chaque transport, les poly-robots peuvent être reconfigurés pour s'adapter au type de charge. La capacité du poly-robot dépend de la configuration et du type de charge. On note p-bot une configuration avec p robots élémentaires. Notez qu'un 1-bot est un mono-bot qui travaille seul. L'objectif, dans un premier temps, est de déterminer le nombre de robots élémentaires nécessaires pour déplacer un ensemble de charges dans un horizon de temps spécifié au coût minimum. Dans un second temps, le but est de minimiser le temps de transport de toutes les charges avec un nombre de robots élémentaires donné.

Organisation du manuscrit et contributions

Le chapitre 1 présente le contexte dans lequel s'inscrit la thèse. Il s'agit tout d'abord de présenter les entrepôts et l'automatisation du stockage et du transport. En outre, une attention est portée aux systèmes multi-robots, plus précisément aux systèmes coopératifs et reconfigurables. Une revue de la littérature est donnée pour le problème de dimensionnement d'une flotte. Dans le cadre des véhicules autonomes, le problème de dimensionnement d'une flotte n'a été considéré ni pour les robots coopératifs ni pour les robots reconfigurables. Les chapitres suivants de cette thèse comblent ce vide.

Dans le chapitre 2, nous considérons le transport de charges identiques par des robots coopératifs et non coopératifs ayant une capacité supérieure à un. Un cadre mathématique déterministe est développé pour le dimensionnement d'une flotte de robots identiques, nommés 1-bots, qui ont la possibilité de coopérer. Un ensemble de p 1-bots qui coopèrent sur une tâche donnée constitue un p-bot. Lorsque la coopération n'est pas autorisée, nous obtenons une expression sous forme analytique pour le nombre optimal de 1-bots. Lorsque la coopération est autorisée et la reconfiguration interdite, nous formulons le problème de dimensionnement de la flotte par un programme mathématique sous la forme d'un PLNE (programme linéaire en nombres entiers). Notre modèle mathématique permet de déterminer le nombre de robots à coopérer pour un coût de transport minimal. Dans le cas où la capacité de p 1-bots est inférieure à la capacité d'un seul p-bot, nous conclurons que l'utilisation soit exclusive de p-bots, soit d'un mélange de p-bots et de 1-bots, entraîne une diminution des coûts. Dans l'autre cas, il est optimal d'utiliser exclusivement des 1-bots.

Dans le chapitre 3, nous considérons la possibilité de reconfigurer les robots pour un transport des charges hétérogènes. Deux PLNEs sont écrits pour comparer le coût d'une flotte de robots avec possibilité de reconfiguration et sans cette possibilité. Ensuite, nous étudions un cas particulier avec deux types de charges et deux configurations autorisées (1-bot et p-bot avec $p > 1$). Pour ce cas particulier, des expressions de forme analytique sont dérivées pour le nombre minimum de robots élémentaires avec ou sans reconfiguration. Un deuxième cas particulier avec des capacités unitaires est également étudié. Pour ce cas, nous dérivons des expressions de forme analytique pour le nombre minimum de robots jusqu'à trois types de charge avec reconfiguration, et pour n'importe quel nombre de types de charge sans reconfiguration. Enfin, nous comparons les straté-

gies avec ou sans reconfiguration. Nous montrons que la reconfigurabilité peut diviser le nombre minimum de robots élémentaires jusqu'à un facteur K (avec K le nombre de types de charge). Pour les deux cas particuliers, nous montrons que le gain en nombre de robots est limité mais peut être significatif pour les petites flottes. Enfin, dans une variante où la demande est par période de temps et non sur tout l'horizon temporel, nous montrons que le gain en nombre de robots peut être très important.

Dans le chapitre 4, nous montrons que le problème est fortement NP-difficile. Dans trois cas particuliers (une seule période, un seul type de charge ou une seule configuration), le problème peut être résolu en temps polynomial avec des algorithmes de programmation dynamique appropriés. Nous dérivons ensuite de nos résultats théoriques un algorithme heuristique efficace pour le cas général. Une étude numérique montre que l'algorithme heuristique peut être appliqué avec succès même pour de grandes instances et a de bonnes performances sur les instances testées.

Dans le dernier chapitre 5, nous considérons le problème d'ordonnancement pour une flotte donnée. L'objectif est de minimiser le temps de transport de toutes les charges. Nous proposons une formulation mathématique du problème qui peut être résolue avec un solveur d'optimisation linéaire. Nous montrons que la reconfigurabilité peut réduire considérablement la durée du transport lorsque les zones de dépôt des charges sont situées à différents endroits.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and state of the art

Contents

This first chapter introduces the reader to the context of the problem and everything that surrounds this thesis. First, we present how warehouses work, then we talk about their automation. Finally, we focus on the core of this thesis: the latest concepts of cooperative reconfigurable robots and the fleet problems. The chapter provides an overview of the current status of the subject of warehouses and the importance of the issue of warehouse automation, which has seen rapid progress recently. Despite the convenience and benefits of automated storage and retrieval systems, they remain expensive and fixed. Robotic systems can be quickly adapted to customer needs and variable flows of products to transport, and cooperation between them promises great prospects. However, in the context of autonomous vehicles, the fleet sizing problem has not been considered either for cooperative robots or for reconfigurable ones. This dissertation fill this gap.

1.1 Warehouses

With the increasing development of e-commerce, the number of warehouses is increasing every year. For example, Figure 2 shows that the number of warehouses in the U.S. increased by 31% between 2007 and 2020, representing 4587 new warehouses.

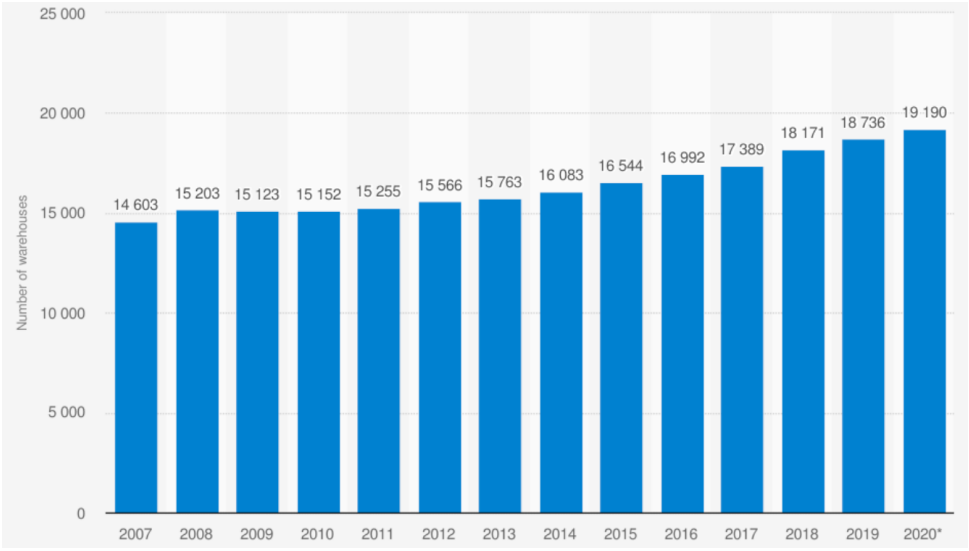


Figure 2: Total number of warehouses in the United States 2007-2020 according to Statista [Mazareanu, 2021]

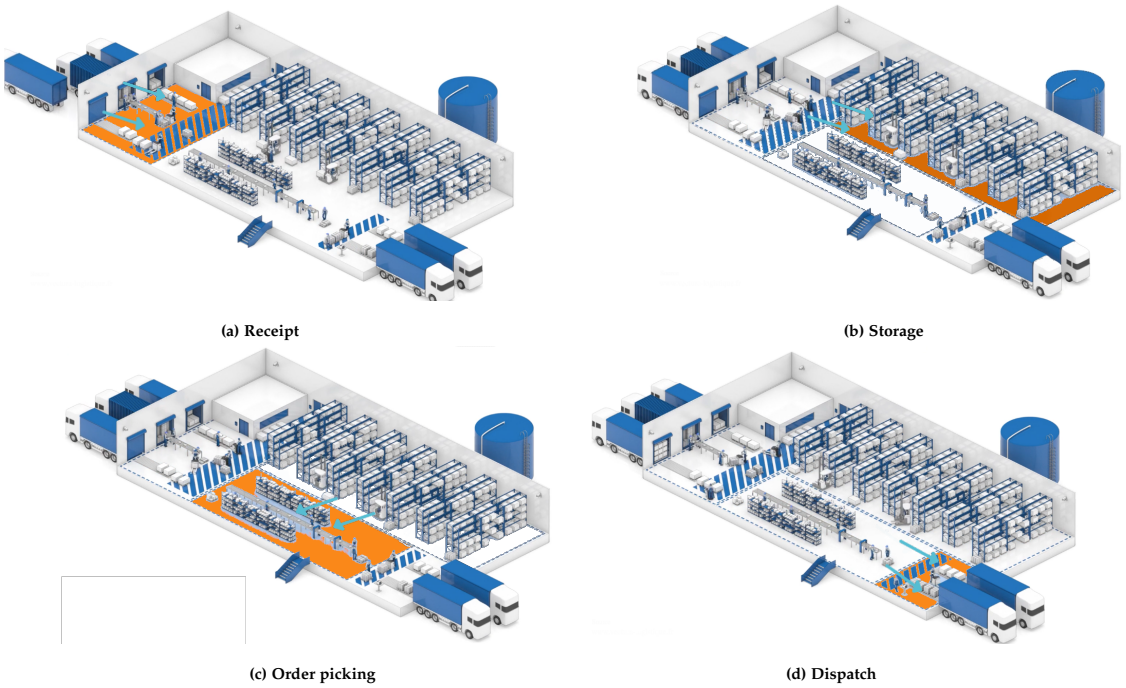
1.1.1 Definition and functions

According to van Geest et al. [2021], a warehouse is a building intended to store goods for commercial purposes. The main functions include receiving, storage, order prepara-

tion and dispatching. Each function corresponds to an area in the warehouse (in orange in Figure 3). Loads (such as pallets, boxes, etc.) are transported between the four zones. Receiving and dispatching are the warehouse's interfaces to the outside world for inbound and outbound material flow, respectively. Incoming goods are unloaded from transporting vehicles (truck, wagons, planes, ships...) and then stored in the storage area.

Different storage strategies can be used such as random storage or storage based on the physical characteristics of the goods (storage on pallets, in boxes, etc.). The warehouses are composed of a storage area which can be made up of two parts: the reserve area, where the products are stored in the most economical way (bulk storage area) and the gripping area where the products are stored for easy retrieval by an order picker. As the stock in the gripping area is depleted, new products are then transferred from reserve storage to the gripping area [Cormier and Gunn, 1992].

The verification and supply of goods are considered separate functions from reception by the company Mecalux. Mecalux [2016] define the objective of a warehouse as being the regulation of the differences between the input flows (what is received from suppliers, production plants, etc.) and those of output (the products sent in manufacturing centers, at points of sale, etc.).



(a) Receipt (b) Storage

(c) Order picking (d) Dispatch

Figure 3: Main functions of a warehouse [VecturaLogistique, 2021]

A complete overview of characterization of warehouses along the views processes, resources and organization is provided in Rouwenhorst et al. [2000].

1.1.2 Storage units

According to Rouwenhorst et al. [2000], the storage unit is a volume in which products can be stored. Examples of storage units are containers, pallets and boxes (cardboard boxes or plastic boxes).

Boxes or bins are generally made of cardboard or plastic (Figure 4). They preserve the integrity of the product and protect it.

(a) Cardboard boxes

(b) Plastic boxes

Figure 4: Cardboards and plastic boxes stored on pallets [Free3D, 2023]

The standardization of the storage units allows, on the one hand, to simplify the loading of the pallets in trucks whose dimensions are also standardized, and on the other hand, to put on the pallets a multiple of the number of standard boxes (Figure 5).

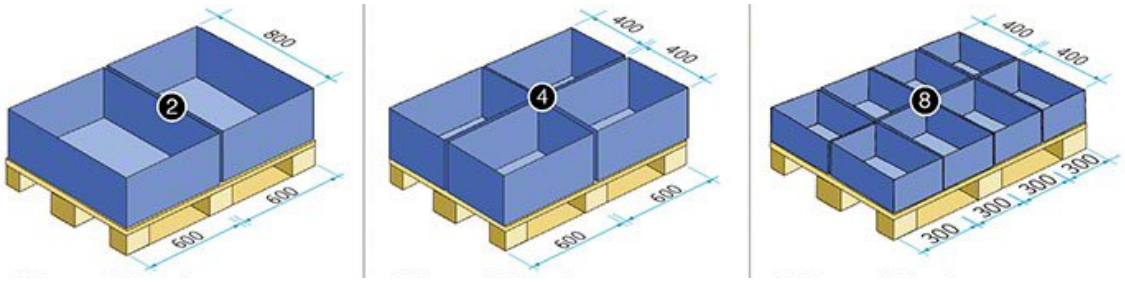


Figure 5: Boxes on pallets [Mecalux, 2016]

"A pallet is a rigid horizontal platform, of minimum height compatible with handling by means

of pallet trucks and/or forklifts or other suitable equipment, used as a support for the gathering, loading, storage, handling, stacking, transporting or displaying of goods and loads" [ISO 445:2013, 2013]. The first standardization of pallets was proposed with dimensions (see Figure 6) of:

- 1200mm x 800mm x 144mm;
- 1200mm x 1000mm x 144mm.

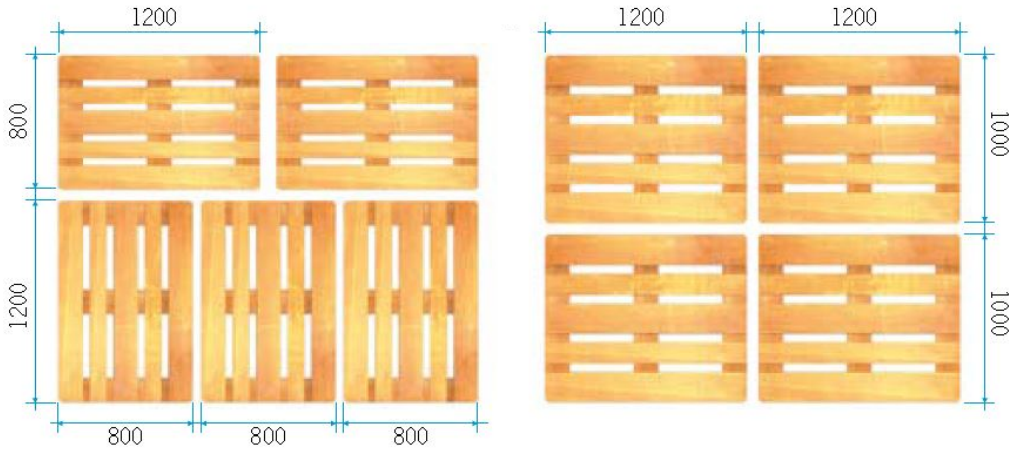
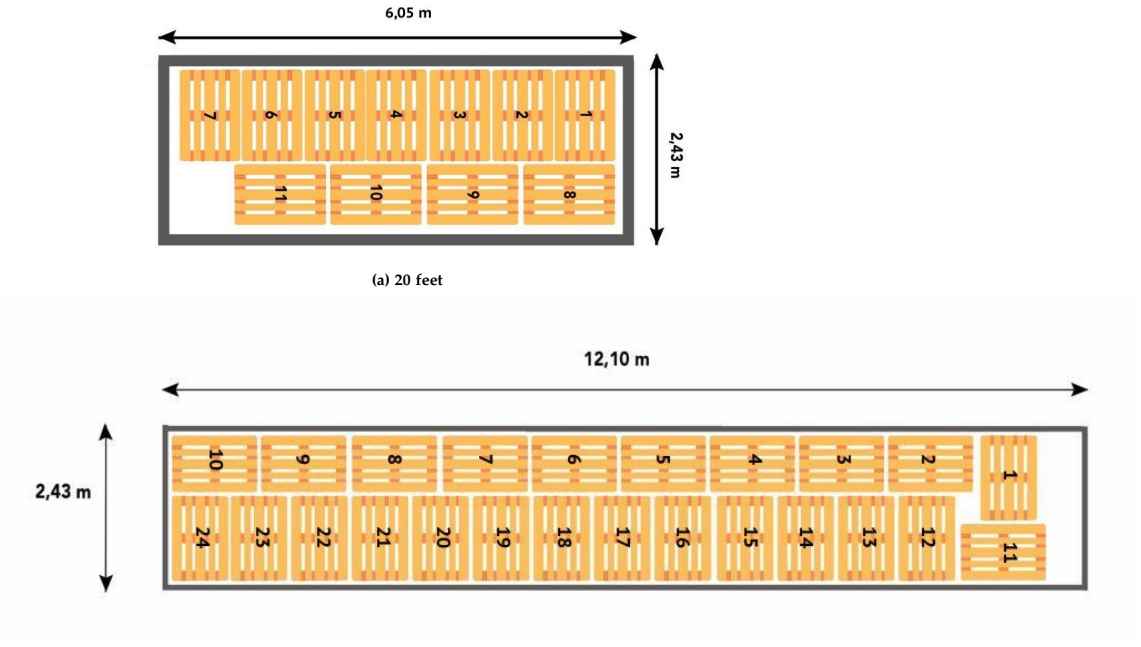


Figure 6: Dimensions of pallets [Mecalux, 2016]

The International Organization for Standardization (ISO) proposes 6 different standard dimensions of pallets depending on the country. In Europe, the UNE-EN 13698-1 standard specifies the manufacturing characteristics of these supports under the names of europallet or EPAL with the dimensions of 1200 x 800 mm. An europallet weighs about 25 kg and can support loads of up to 1500 kg. To note that the pallet height, which is defined by the user, depends on the masses and the strength of the stored boxes. It rarely exceed 2,5 m in height (pallets are often filmed) to limit the risks of pallets warping in curves during road transport.

A container is built for intermodal freight transport, meaning these containers can be used across different modes of transport – from ship to rail to truck – without unloading and reloading their cargo [Lewandowski, 2016]. The containers of 8 feet (2,44 m) wide, and of either 20 or 40 feet (6,10 or 12,19 m) standard length, are defined by ISO standard [ISO 6346:1995, 1995]. In a 20 feet container, it's possible place up to 11 EPAL pallets (see Figure 7a) on the ground, in a 40 feet - up to 24 EPAL pallets (see Figure 7b).



(a) 20 feet

(b) 40 feet

Figure 7: Container loading plan with EURO pallets [MouvBOX, 2023]

The Table 1 provides a summary of the capacity of containers and pallets described above.

	EPAL
Container of 20 feet	11
Container of 40 feet	24

(a) Containers

	Box 0,8x0,6 m	Box 0,4x0,6 m	Box 0,4x0,3 m
EPAL	2	4	8

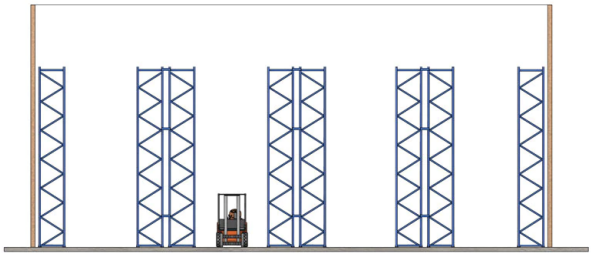
(b) EURO Palettes

Table 1: Capacity of containers and pallets

1.1.3 Shelving

A distinction is made between single and multiple depth storage systems. For simple storage systems, a shelf depth is a pallet length (Figure 8). Storage capacity can be doubled by using double-depth racks (Figure 9). The double depth saves space by eliminating the aisle that initially separated the racks. On the other hand, only the access to the first pallet is direct. To be able to reach the second pallet, the first one has to be extracted.

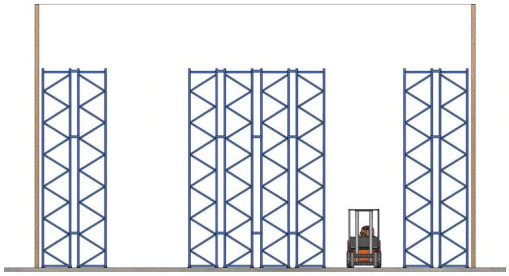
Regarding the dimensions of the shelves: the depth of the shelf is generally equal to the



(a) Single depth [Mecalux, 2016]

(b) Industrial example Groupe HBF

Figure 8: Single depth configuration



(a) Double depth [Mecalux, 2016]

(b) Industrial example COLBY

Figure 9: Double depth configuration

length of the EPAL. It means EPALs are inserted along the long side of the pallet; the width of the shelf is a multiple of the width of the pallet (2, 3, 4, 5 spaces). To note that the height of the racks is 2 m, 6 m or 12-13 m, rarely more.

The aisles are generally 3,5 m wide to allow a forklift to make a quarter turn in order to position its fork in front of the shelving. Aisles can be one-way or two-way. An aisle of 3,5 m can be two-way (enough for the crossing of two 1, 5m wide vehicles).

1.2 Warehouse Automation

Despite the rapid growth of progress, there are still many manual warehouses where all manual tasks are performed by humans: unloading trucks, content checking, horizontal and vertical transfers, unit picking, order preparation, packaging. These are the very laborious and repetitive tasks, so it is not surprising that logistics warehouses are not attractive for human workers and get increasingly robotized.

Roland Berger [2016] anticipates that 1,5 million jobs will be replaced by robots between 2016 and 2026 in the Eurozone and that handling costs will be reduced by 20 to 40% in the same time thanks to robotic solutions. Companies decide to buy a robot by com-

paring the cost of its use with that of an operator, knowing that a machine is generally amortized over 3 years. Figure 10 shows the evolution of the hourly cost for robots and human operators. We see that robots are becoming more and more affordable. A recent study [Barosz et al., 2020] confirms the interest of automated systems compared to operators in a certain number of situations as well as the efficiency of the use of automated systems compared to operators.

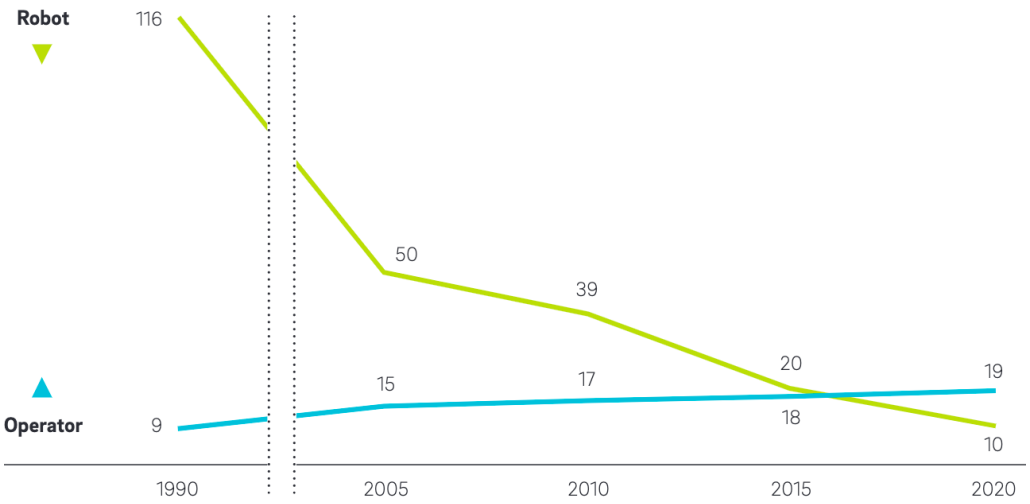


Figure 10: The hourly cost of robots and human operators in France in €/hour [Roland Berger, 2016]

1.2.1 Automated storage

Automated storage systems have the greatest capacity, storage density and, therefore, require high investments. The principle of operation is as follows: loads arrive, the system chooses the location to store the load, an automated system grips the load and transports it to an empty slot in the storage space.

Existing classifications are rather based on machine elements (carousel, shuttles, etc.) than on mechanical or geometric properties. We distinguish three systems (Figure 11): carousel, VLM (Vertical Lift Module) and AS/RS (Automated Storage and Retrieval System). Automated storage systems could be classified by storage thickness. For example, VLM and carousels belong to the category of single thickness systems whereas AS/RS sometimes allow storage in multiple thicknesses.

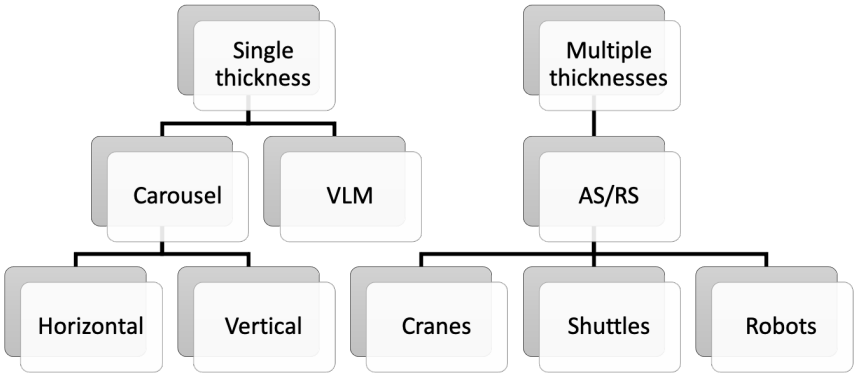


Figure 11: Types of automatic storage

Automated Storage and Retrieval System

An automated storage and retrieval system typically consists of racks serviced by cranes traversing the aisles between the racks. An AS/RS is able to handle loads without operator intervention, so the system is fully automated [Roodbergen and Vis, 2009]. AS/RS can use pallets or boxes [SSI Schaefer, 2023].

There is a wide variety of load recovery instruments. To do this, cranes, shuttles and robots are used, individually and in combination:

- *Cranes*

  Cranes can move in the same plane, horizontally and vertically, usually simultaneously. Note that two cranes can move on the same horizontal rail to serve different parts of the stock. The vertical movements can be done in parallel without problem. Figure 13a shows the system proposed by RINAC, in which such a crane is represented in orange. The capacity of systems with cranes is limited, as only one crane can move horizontally at a time. This led to a new generation of grips, the shuttles.

- *Shuttles*

  A shuttle is a mobile platform that moves inside the AS/RS. The shuttle system uses elevators to move the shuttles between levels, in turn the shuttles can move on rails along the X and Y axes. In Figure 13b, orange shuttles can be seen over the entire grid used by Vanderlande. A similar method is used in systems with robots that can also, among other things, move on the floor outside the storage system.

- *Robots*

A robot is a machine that senses, decides and acts autonomously. A robot must have sensors necessary to obtain information about its environment [Bekey, 2005]. We give two examples of the use of robots with gripping systems.

The first example of a system with robots that got a lot of attention is SqUID made by BionicHIVE (Figure 12a). The SqUID consists of a synchronized autonomous robotic fleet. The robots move vertically and horizontally in tracks attached to racks, which can cause traffic jams. The SqUID is only fixed on a single rack, which facilitates the installation but exposes to a mechanical weakness of the guides. Exotec's Skypod robots (Figure 12b) move horizontally on the ground and hoist themselves vertically by resting on the racks located on either side of the aisle. This double support avoids lifting the cantilever load (robustness) but imposes a good parallelism of the slides, which requires a flat and rigid slab [EXOTEC, 2023].

An integrated control system and intelligent real-time data analysis allow the algorithmic engine to dynamically learn from problems created in a warehouse and apply resolutions to all warehouses in the network.

(a) SqUID [BionicHIVE, 2023]

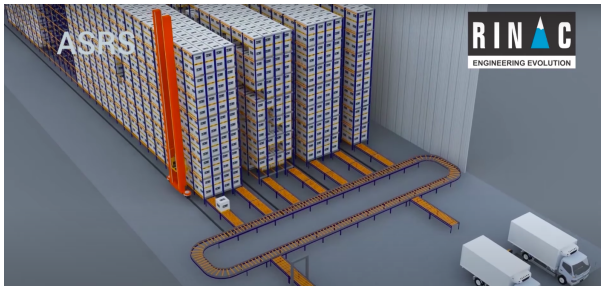
(b) Skypod [EXOTEC, 2023]

Figure 12: Robots

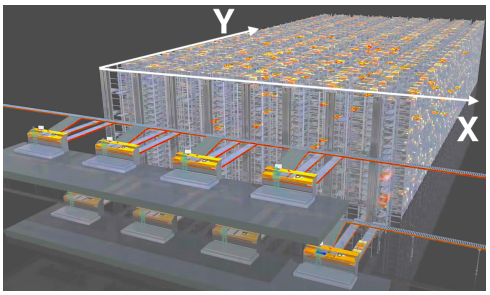
Another example are RAFT (Right Angle Fast Transfer) robots with VTU (Vertical transfer units). RAFT robots move in the same way as shuttles on the X and Y axes, and VTUs allow them to move in height (Figure 13c). Such systems are sometimes referred to as AVS/RS (Autonomous Vehicle Storage and Retrieval System) [Lenoble, 2017].

Another category is RCSR systems (Robot-Based Compact Storage and Retrieval

Systems), robots work only at the highest level and are suitable to take the load without moving in height. AutoStore, developed by Hatteland (Figure 13d), is the first implementation of the RCSR system [Azadeh et al., 2017].

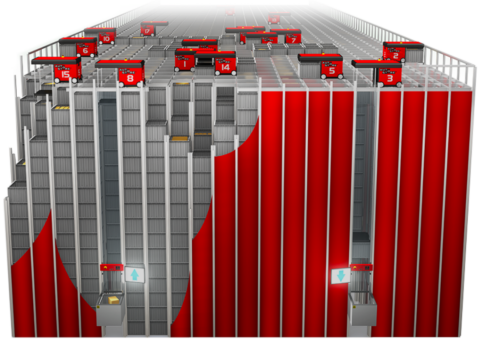


(a) System with a crane (source: RINAC)



(b) System with shuttles (source: Vanderlande)

(c) RAFT system with robots and VTU (source: SRSI)



(d) RCSR system (source: Hatteland)

Figure 13: Examples of AS/RS

Vertical lift module

The vertical lift modules consist of two columns of trays with a mechanical inserter/extractor positioned in the center (Figure 14). The inserter/removal moves up and down between stored trays, automatically locating and retrieving them as needed, like an elevator with doors that open both front and back [Kardex, 2021a].

VLMs are generally used in groups of several machines to perform entries or exits from a tray of a given VLM during the collection of articles on another VLM [Lenoble, 2017].

Carousel

Carousels are automated storage and retrieval systems in which shelves are linked together and rotate in a closed loop. The rotation is either horizontal or vertical. In this system, the picker has a fixed location in front of the system and the system transports the items to the picker [Azadeh et al., 2017].

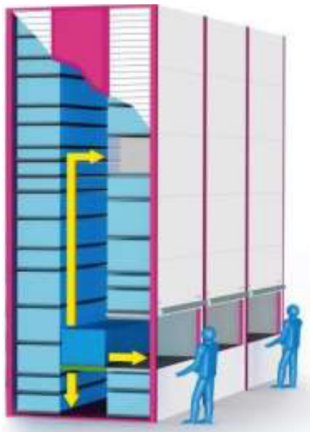
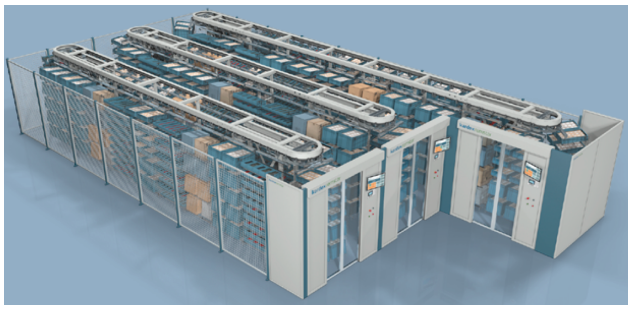


Figure 14: VLM [Lenoble, 2017]

The *vertical* carousel modules are a series of supports attached to fixed locations to a chain drive (Figure 15a). The movement is powered by a motor, which sends the carriers in a vertical loop around a track in both forward and reverse directions - similar to a Ferris wheel. Goods are stored or retrieved through an ergonomic access opening with a work counter [Kardex, 2021b].

(a) Vertical



(b) Horizontal

Figure 15: Carousels [Kardex, 2023]

The *horizontal* carousel modules consist of an oval rail supporting rotating bins with shelves (Figure 15b). A motor located inside the oval track propels the transporters around the track horizontally, stopping at a pre-determined access point for cargo storage or retrieval [Kardex, 2021b].

Carousels are effective in increasing storage density and facilitating access to products, often stored in boxes. On the other hand, the mass of the carousel limits the accelerations. The space between the links of the chain is constant. Care must also be taken to balance the system. VLMs allow varying storage heights and have fewer balancing issues. In both cases, access to the product is at a single point in the system, often on

the ground, and there is a variable waiting time before recovery.

Despite all the advantages of automated systems, they are still expensive to install and once installed they are stationary, which is a major drawback in often reconfigurable warehouses.

1.2.2 Automated transport

For transporting loads in warehouses, different types of automated vehicles are used. Several terms are used to designate these vehicles: Automated Guided Vehicle (AGV), Autonomous Mobile Vehicle (AMV) and Autonomous Mobile Robot (AMR), Mobile Robot and Mobile Manipulator Robot. Some of these terms are interchangeable. For clarity, we will divide vehicles into three categories:

Automated Guided Vehicle

AGVs are vehicles guided on a predefined path, often optical or magnetic strips stuck to the ground. These vehicles generally stop when they encounter an obstacle on the road. According to Digani [2016], AGVs are used for transporting loads from one warehouse area to another (Figure 16) and consist of the following elements: a localization system (usually laser systems or magnetic systems), a safety system (proximity sensors or bumpers to avoid collisions) and a communication system (most of the type between AGVs and a supervision station, sometimes between AGVs).

Figure 16: AGVs used for automated goods transport in a warehouse [Cardarelli et al., 2017]

Mobile robots

A mobile robot has a certain degree of autonomy greater than an AGV. The concepts of AMV and AMR therefore fall into this category. A complete overview of the last decade of these technologies can be found in [Oyekanlu et al., 2020]. AMVs are similar to AGVs but in addition have the ability to avoid obstacles: in the event of an obstacle in its path, an AMR is able to propose a bypass strategy to finish its task [Andersson, 2022]. An AMR is also often considered an advanced AGV. For example, Andersson [2022] considers that a main function of an AMR is to bring a load (pallet, box, etc.) from a storage area to a picking station, to deposit its load and return to storage. The author considers an AMR only as a basis for transport. At the same time, robots that are able to lift a load (a pallet, a trolley, a roll, a shelf, etc.) and move it, have their name: turtle robots or MRFS (Mobile Robot Fulfillment Systems) which move loads carrying it on their backs. These robots are respectively used in an RMFS (Robotic Mobile Fulfillment System), which according to Azadeh et al. [2017], has three main components:

- Robot Drive Units: these robots receive instructions from the central computer to transport inventory shelves to the workstation for restocking or picking. Nowadays, there are also decentralized controlled systems.

- Inventory Shelves: the shelves are mobile industrial shelves that hold stored items. The small shelves are used for weights up to 450 kg and the large shelves are used for weights up to 1300 kg.

- Workstation: ergonomically designed areas where human workers perform shelf restocking, picking, and packing functions.

RMFS was patented by KIVA Systems Inc. [Mountz et al., 2008], which was later acquired by Amazon in 2012 and renamed AmazonRobotics.

Figure 17 shows a RFMS, where a robot brought products to a worker. This system allows mobility in the plan but is limited in height and mass of shelves.

Another example of a mobile robot, described by Urru et al. [2018], is the SOTO manufactured by Magazino (Figure 18a). The SOTO is a mobile robot that performs industrial material supply: the robot enables efficient automated driving supply.

Figure 17: RMFS [Amazon Robotics, 2023]

In this category of mobile robots, we can also include automated carts. For example, Balyo automates forklifts so that they are able to move without an operator: The TRUCKY robotic pallet truck based on a manual pallet truck (Figure 18b). It is capable of carrying two pallets at a time, and up to 3000 kg, it can perform platform loading and unloading, long-distance transfer and stock line pick-up /drop-off.

(a) SOTO [MAGAZINO, 2023]

(b) TRUCKY [BALYO, 2023]

Figure 18: Examples of mobile robots

Mobile manipulator robot

A mobile manipulator robot consists of a mobile base supporting one or more robotic arms. A fixed base manipulator arm has a certain working space in which its accuracy

and speed can be characterized [Gifford, 2006]. The development timeline of mobile manipulator robots is shown in Figure 19.

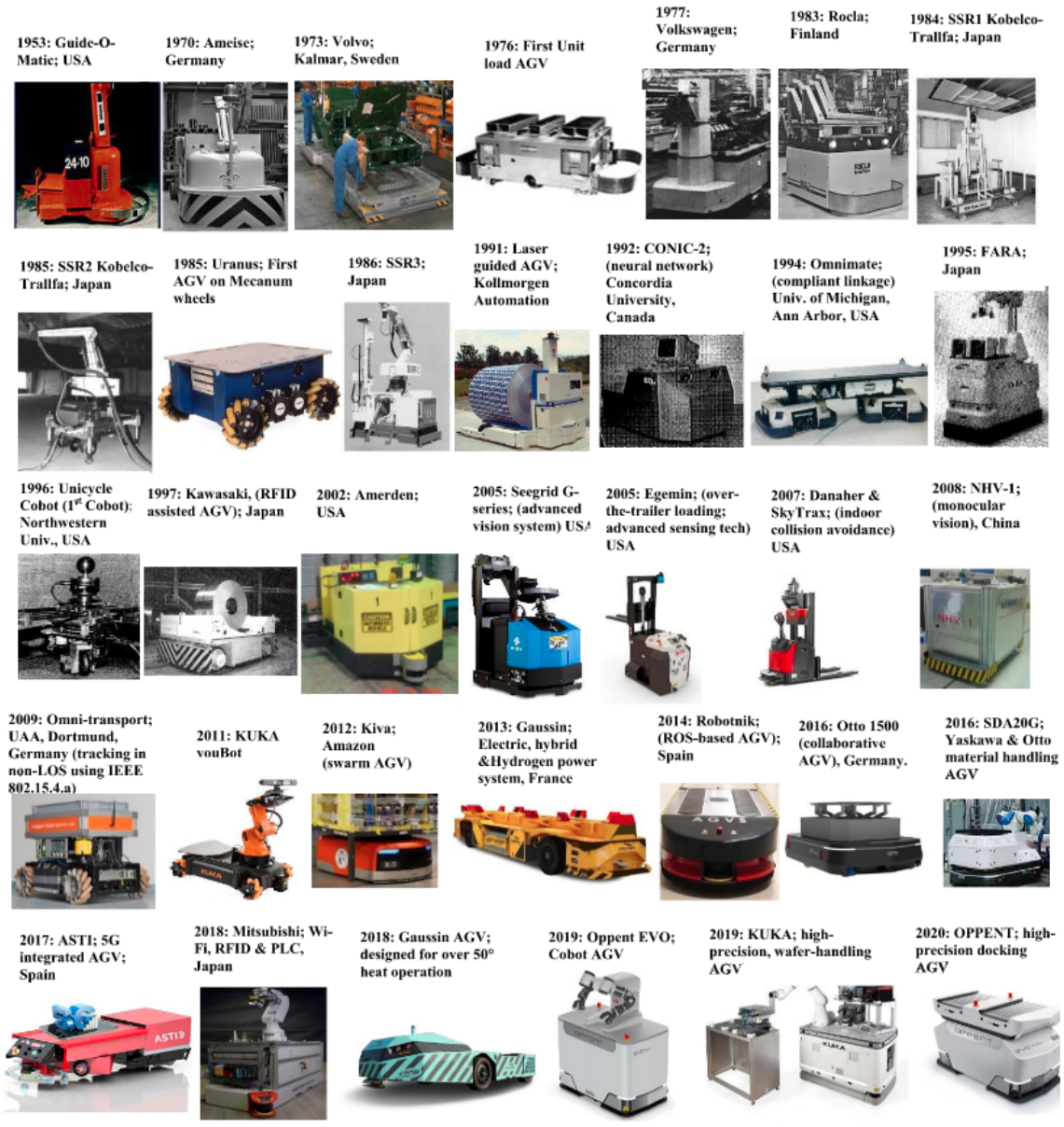


Figure 19: Development timeline of mobile manipulator robots [Oyekanlu et al., 2020]

Robots are increasingly used for Small Load Carriers (SLC) [Urru et al., 2018]. In this regard, Urru et al. [2018] considers these new means of transport highlighting their most important characteristics:

- Navigation in a dynamic environment;
- Transport of a set of SLC;
- SLC handling (full-empty SLC exchange);

- Great maneuverability.

Figure 20 shows an example of AMADEUS (Autonomous Manipulator Device for Strengthening Manufacturing in Europe) robot. The platform is a combination of an AGV with an application-specific configuration and a robotic manipulator.

Figure 20: AMADEUS demonstrator in the industrial scenario, by Fraunhofer [Urru et al., 2018]

1.3 Multi-robot systems

A system composed of more than one robot is called a Multi-Robot System (MRS) [Gautam and Mohan, 2012]. According to Yan et al. [2013], multi-robot environments can be cooperative or competitive.

We speak of cooperative behavior when robots interact with each other in common interests. For example, search and rescue of people [Balakirsky et al., 2007] or transport of loads [Yan et al., 2012]. One of the first papers on cooperation between robots was done by Alami et al. [1998]. The authors considered the autonomous coordination of the robots for transporting goods. The central station only sends high level missions. It is then up to the robots to refine, plan and coordinate route sections and crossings use, as well as trajectories in open areas.

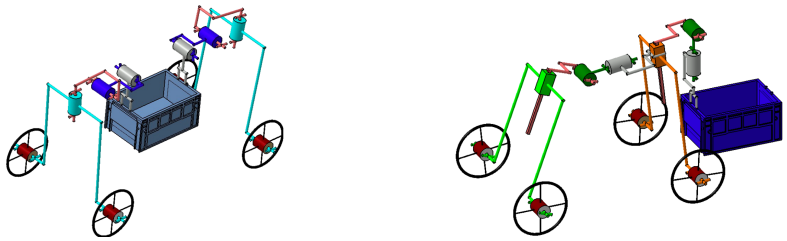
The competitive environment means that the robot only works in its own interest. A typical example is the game of chess. A mixture of cooperative and competitive envi-

ronments can be found in the robot soccer league, where we see cooperation within a team and competition between teams [RoboCup, 2021].

1.3.1 Cooperation

Cooperative robots capable of working in parallel on the same task open wide perspectives [Noreils, 1992]. Cooperation can be defined as the joint performance of a task [Jung et al., 1998]. For Tuci et al. [2018], there is cooperation if the task cannot be performed sequentially by a single robot and requires coordination of actions and communication between robots. Communication is explicit when the robots communicate directly with each other, and implicit when they communicate through an object which is, for example, transported.

There is different connection modes, for example, a large load can be transported by several small robots connected to the load (co-manipulation mode) or one robot can transport the load while connecting to another robot to increase stability (connection mode) [Chebab, 2018] as illustrated in Figure 21.



(a) co-manipulation mode (b) connection mode

Figure 21: Different modes of cooperation Chebab [2018]

Whatever the connection mode, there is coordination between robots, that is to say an exchange of information in order to be able to carry out a task together. Coordination is the basic process allowing robots to collaborate with each other [Farinelli et al., 2004]. Different methods and algorithms are developed so that the robot can take into account the actions of other robots in the system, for example, Majcherczyk [2020] focus on how to enable the exchange of information for robots to gain better knowledge of the global state by building a collective semantic map from aggregated information. Coordination can also include a teamwork strategy that allows a set of robots to solve the problem of movement between a starting point and an objective through a safe path [Chaves Osorio, 2021] or the movement of a load from its arrival location to the unloading point, when

the location of the load is not known at advance [Nath and Niyogi, 2021].

Concerning the cooperation for the transport of objects, three categories of cooperation can be distinguish [Tuci et al., 2018]:

- Pushing-only strategy: robots are not physically attached to the object, and transport is achieved by pushing the object. This method can be used when robots cannot pull an object. Kube and Zhang [1993] has done pioneering work in this area, demonstrating the ability to move an object without direct communication between robots. A more complex model with obstacles is presented by Wang and De Silva [2006].

- Grasping strategy: robots are physically attached to the object, and transport can be achieved by pushing or pulling (or both) the object. This strategy assumes that the robots are equipped with a gripping tool. Algorithms have been developed to find the position of the robots to ensure maximum stability of the robots carrying the load [Sasaki et al., 1995; Hichri et al., 2016].

- Caging strategy: robots surround the object and block it during transport, unlike the pushing-only strategy. It is essential to position the robots correctly according to the shape and size of the object, so that the object does not escape from the cage [Campos and Kumar, 2004].

A complete review of cooperative MRS can be found in Rizk et al. [2019].

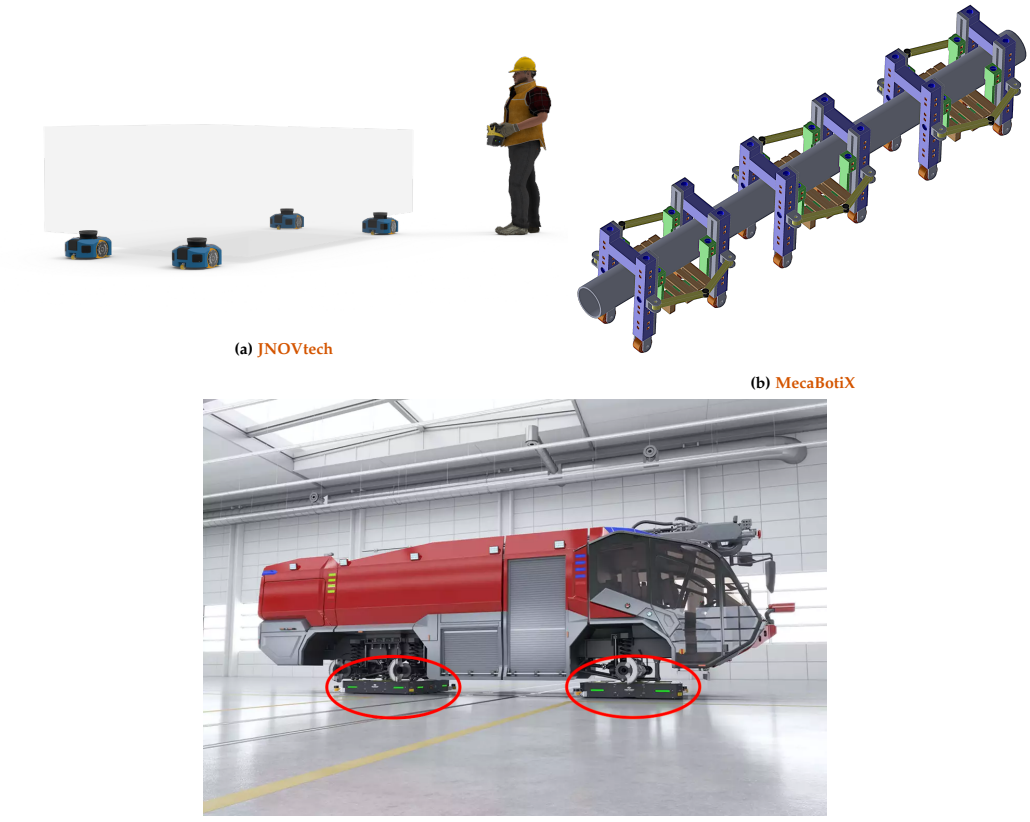
1.3.2 Reconfiguration

A reconfigurable robotic system is an assembly of modules that can attach and detach from each other to modify and adapt to different tasks and environments [Bojinov et al., 2000]. The ability of individual modules or robots to be connected in different ways to perform the required task offers promising potential [Arai et al., 2002]. Stoy et al. [2010] define three categories of self-reconfigurable robots based on the number of modules: *pack* robots, *herd* robots and *swarm* robots. Pack robots are composed of several modules, usually in the range of tens, and require strict coordination due to the fact that the individual modules play a crucial role in the robot's overall performance. Herd robots are composed of a large number of modules, usually in the range of hundreds, and

global coordination of these modules is challenging. They are better managed as a collection of groups since the actions of individual modules are still significant but not as critical to the overall performance of the robot. Eventually, swarm robots are comprised of countless modules. Here, each module is controlled locally since the impact of an individual module on the overall behavior of the robot is minimal.

Several prototypes of reconfigurable robotic systems have been developed. The reader is referred to Jahanshahi et al. [2017] or Seo et al. [2019] for a survey on this topic. For instance, systems using multiple modules can create different forms to perform different tasks: it could turn into a snake to reach into narrow places, into a hexapod to carry a load or it may split into many smaller robots to perform a task in parallel [Castano et al., 2000; Yim et al., 2000]. The self-reconfigurable robots can also be used as conveyors. The spherical shape of the ATRON modules enables them to function as wheels, facilitating the construction of surfaces that have the capability to transport items [Østergaard et al., 2006; Brandt et al., 2007]. Shen et al. [2006] demonstrate a solution based on SuperBot modules that can perform multimodal locomotions such as snake, caterpillar, insect, spider, rolling track, H-walker, etc. Chebab [2018] focuses on the design of new architectures of modular mobile manipulators that can cooperate with each other to perform tasks in industrial or service contexts concerning the handling and transport of boxes. Another application of reconfigurable robots can be found in Mars exploration, where tasks such as transportation or building construction have to be performed with limited resources [Irawan et al., 2019]. A survey of modular system for multifunctional applications in space exploration is presented by Post et al. [2021].

In commercial products, there are several examples of technical solutions for cooperative and reconfigurable robots (Figure 22). JNOVtech robots connect directly to the load (Figure 22a) and MecaBotiX robots (Figure 22b) can connect to each other and /or to the load. Another example of robot cooperation is presented by the Strothmann Round-Track, mobile platforms that cooperatively move big loads on rails (Figure 22c).



(a) JNOVtech

(b) MecaBotiX

(c) Strothmann RoundTrack

Figure 22: Robot technical solutions

1.4 Fleet management

A warehouse of a manufacturing company today is characterized by dynamic production processes governed by the demands of a rapidly changing global economy, such as the increasing number of product variants, customization of products and responsiveness to changing market conditions [Urru et al., 2018]. In order to be competitive, companies are forced to seek the advantageous solutions, specially for vehicle fleet management. Fleets of vehicles require solving several problems related to both the internal management of this fleet and global issues such as determining the required number of vehicles. Rjeb [2022] identifies several of the fleet management problems:

1. Scheduling problem

   A scheduling problem consists in assigning tasks (e.g. production or transportation) to resources (e.g. machines or vehicles) with the goal to minimize one or more objectives (e.g. makespan that is the time to complete all transportation

tasks) [Pinedo, 2012]. Job shop problem is often considered. This problem consists in multiple jobs which are processed on several machines. Each job contains of a sequence of tasks, which must be performed in a given order, and each task must be processed on a specific machine. The job-shop scheduling problem with one mobile robot is investigated by [Hurink and Knust, 2005; Caumond et al., 2009] while the problem with several mobile robots has been tackled by [Deroussi et al., 2008; Lacomme et al., 2013; Baruwa and Piera, 2016; Fontes and Homayouni, 2019; Yao et al., 2023]. Most of the job-shop scheduling problems with mobile robots are NP-hard. Hence the exact approaches, such as Mixed Integer Linear Programming (MILP), work only for small instances [Caumond et al., 2009; Fontes and Homayouni, 2019; Yao et al., 2023]. Approximate solution methods have also been considered to tackle larger instances [Deroussi et al., 2008; Lacomme et al., 2013; Baruwa and Piera, 2016]. We refer the reader to Yao et al. [2023] for a detailed review of these works.

2. Vehicle routing problem (VRP)

   A VRP consists in determining the optimal route to minimize the cost of transport. Numerous articles define the limits of the problem formulation in different ways. According to Golden et al. [1984], the problem involves a predetermined number of vehicles, with the same capacity. A key assumption is that fixed (or acquisition) costs have already been incurred and only variable (or routing) costs need to be explicitly considered. The objective is to minimize the total transport cost which is a function of the total distance traveled by the fleet of vehicles. In the article of Bodin and Golden [1981], this problem is defined more broadly and, in addition to the above objective, it can have objectives such as: minimizing the sum of fixed and variable costs or minimizing the number of vehicles required. Sitek and Wikarek [2019] describe various versions of the problem. A large amount of literature has recently been devoted to the environmentally friendly type of vehicles. A complete review of the literature on this topic can be found in Asghari et al. [2021].

3. Fleet sizing problem (FSP)

   Determining the appropriate fleet size, which refers to the number of vehicles to acquire or lease in order to meet demand, often takes precedence and heavily in-

fluences the routing decision [Golden et al., 1984]. The optimal number of vehicles is considered for various objective functions. Most of the time, the objective is to minimize the number of robots required to achieve a set of transportation jobs in a time interval. Several works consider more elaborate objective functions. Beaujon and Turnquist [1991] maximize the total profit (difference between revenues and total transportation costs, including penalty costs for unmet demand). Etezadi and Beasley [1983] minimize the cost of a fleet of purchased or leased vehicles. Sinriech and Tanchoco [1992] minimize the cost by applying penalties if performance is not achieved in terms of quality of service. Many articles on the topic of FSP in a road freight transportation are given by Żak et al. [2011].

In what follows, we focus on the fleet sizing problem. This problem is the main one in this dissertation and that is why special attention is paid to it. First of all, we will discuss this problem in the general context of transport systems. Then, the fleet sizing problem for autonomous vehicles is considered. In this area, attention is paid to automated guided vehicles (AGVs) and autonomous mobile robot (AMRs). However, there is a noticeable lack of articles in the literature on the topic of FSP for cooperative and reconfigurable robots. This is a new topic that will be supplemented by this dissertation.

1.5 Fleet sizing

One of the most challenging issues in fleet management is a fleet sizing problem. The main focus of the fleet sizing problem (FSP) is to align supply and demand. It involves determining the appropriate number of vehicles in the fleet to achieve two goals: ensuring the complete fulfillment of the incoming transportation orders and preventing high fixed costs associated with fleet underutilization [Żak et al., 2011].

1.5.1 Transportation systems

The fleet sizing problem in transportation systems consists in determining the optimal number of vehicles for the transport of goods. This is a key logistics problem which concerns all means of transport (air, sea, road, inside warehouses, ...). Baykasoğlu et al. [2019] provides a review of fleet planning problems (including fleet sizing) in transportation systems. In road transportation, the problem involves tanks and rail cars [Sha and Srinivasan, 2016; Milenković and Bojović, 2013; Cheon et al., 2012], trucks [Mohtasham

et al., 2021; Amjath et al., 2022], vehicles [Rahimi-Vahed et al., 2015; Koç et al., 2014; Kumar et al., 2018] and electric buses [Manzolli et al., 2022]. The issue is not only the composition of the fleet but also the choice of the route [Hoff et al., 2010]. For maritime transport, the question of determining the route can be less relevant due to the fact that the route between the point of departure and destination is connected by a straight line, except when there are areas on the way that need to be bypassed or bad weather condition [Romero et al., 2013].

The maritime fleet sizing problem considered by Pantuso et al. [2014] is related to our study. The authors give an example of an objective function and constraints specific to the maritime fleet problem:

$$\min \quad \sum_{v} C_v^F y_v + \sum_{v} \sum_{r} C_{vr}^V y_v x_{vr}$$

subject to :

$$\sum_{r} Z_{vr} x_{vr} - Z y_v \leq 0, \qquad \forall v \in V$$

$$\sum_{v} \sum_{r} Q_v A_{ir} x_{vr} \geq D_i \qquad \forall t, \forall i \in N$$

$$y_v \in \mathbb{N},\ x_{vr} \in \mathbb{N} \qquad \forall i, \forall v, \forall r$$

- y_v represents the number of ships of type $v \in V$
- x_{vr} represents the number of times route r is sailed by ships of type v
- A_{ir} is equal to 1 if route r calls port i, and is equal to 0 otherwise
- C_v^F represents the cost of including a ship of type v in the fleet
- C_{vr}^V represents the cost of sailing route r with ships of type v
- Z_{vr} is the time consumed every time a ship of type v sails route r
- Z represents the total amount of time available for each ship within the planning horizon
- Q_v is the capacity of a ship of type v
- D_i is the demand of port i

The objective function includes two terms, the first of which is responsible for the price of the fleet, and the second for the the price of the distance traveled. We can see a similar objective function in Chapter 2, where the type of robot would correspond to the type of ship. The constraint on the demands is also similar. As for the constraint on the time, in our case, a time horizon is given for which it is necessary to transport all loads, in the case of the model described above, an allotted time is given for each type of ship.

1.5.2 Autonomous vehicles

A particular development over the last decade has taken place for AGVs and AMRs [Oyekanlu et al., 2020], especially in logistics warehouses and industrial production [Andersson, 2022]. Different questions arise when operating a fleet of robots, such as the design of the warehouse architecture [van Geest et al., 2021], trajectory planning with obstacle and collision avoidance [Cardarelli et al., 2017; Lee et al., 2019], service policy [He et al., 2018] and battery charging [Zou et al., 2018]. Due to the fact that autonomous vehicles are expensive, it is crucial to determine the correct type and number of vehicles, which is what we focus on in this section.

We begin the study of this issue by a consideration of sizing methods. Ganesharajah et al. [1998] identify two sizing methods: simulation and analytical method. Simulation can simulate reality with great accuracy and produce the fewest errors, but at an early stage of work it can be difficult to build a model that accurately characterizes the basic properties and necessary parameters. And then an analytical model can be interesting, which makes it possible to obtain an optimal solution and can quickly run through a large number of parameters. Error in performance estimates using analytical models is generally acceptable for the conceptualization phase. The analytical methods is divided into deterministic and stochastic. Vis [2006] highlights in deterministic approach linear programming models that can be utilized prior to the actual operation to estimate the required number of vehicles, but stochastic models, such as queueing networks, aim to incorporate external influences. On one hand, analytical models tend to underestimate the required number of vehicles compared to simulation results. On the other hand, simulation requires a lot of details and hardly copes with large fleets, that's why the literature on the sizing of a fleet of robots is largely devoted to analytical approaches (with stochastic models [Koo et al., 2004; Arifin and Egbelu, 2000] or with deterministic

models [Egbelu, 1987; Rjeb et al., 2021a]).

This dissertation considers an analytical approach, namely deterministic, which is why special attention, in what follows, is paid to deterministic models. The work of Egbelu [1987] proposes four analytical approaches to estimate the number of robots, giving examples for each of them. The author proposes to consider adding dispatching rules and then simulate his models with varying incoming material flow. These models are optimistic according to the author. Rjeb et al. [2021b] refine the results of Egbelu [1987] in the case of homogeneous loads by providing an analytical formula for the optimal number of robots. In the case of heterogeneous loads, the authors formulate the problem as a bin packing problem.

Lee and Murray [2019] investigate a new approach for warehouse order picking. The article focuses on two types of commercially available mobile robots: pickers, capable of grasping items from shelves, and transporters, designed to deliver items from the warehouse to the packing station. The authors determine the optimal combination of picker and transport robots that surpasses the performance of traditional human-based picking operations. Lyu et al. [2019] simultaneously consider the optimal number of AGVs, the shortest transportation time, a path planning problem and a conflict-free routing problem. To study these problems simultaneously, they propose a genetic algorithm combined with the Dijkstra algorithm that is based on a time window. Aziez et al. [2022] focus on the optimization of the number and types of carts and AGVs required to fulfill daily requests in a hospital while optimizing AGVs routes and adhering to time constraints. Each request necessitates specific types of carts, which are transported by the AGVs. The authors present a mathematical formulation and propose a matheuristic approach. This matheuristic leverages a dynamic reoptimization of routes as new requests arrive.

Complete reviews on AGV design/control and on AMR planning/control for intralogistics can be found respectively in reviews in Vis [2006] and Fragapane et al. [2021].

1.6 Research questions

The thesis is in the context of intralogistic of an industrial system comprising several robotic entities, which are mobile and cooperative. The goal is to transport loads from

one zone of warehouse to another zone by reconfigurable robots. A robot is a machine that has a function and can control itself autonomously to perform its function. The reconfigurable robots, named poly-robots, consist of elementary robots that are aggregated in order to jointly perform a task as a single robot. An elementary robot is abbreviated as *bot* and cannot be split into several robots. Each poly-robot configuration has its own transport capacity. The capacity of the poly-robot depends on the configuration and the type of load. We denote by p-bot a configuration with p elementary robots. Note that a 1-bot is a mono-bot that works alone. The time horizon to transport all loads is divided into T periods. At the beginning of each period, the poly-robots are located in the loading area and can be reconfigured. For example, a 3-bot and a 2-bot, i.e. 5 elementary robots, can turn into a 4-bot and a 1-bot. This example is illustrated in Figure 23.

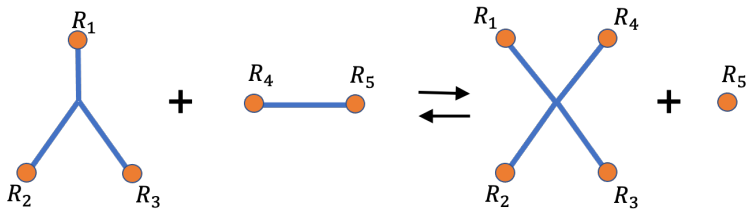
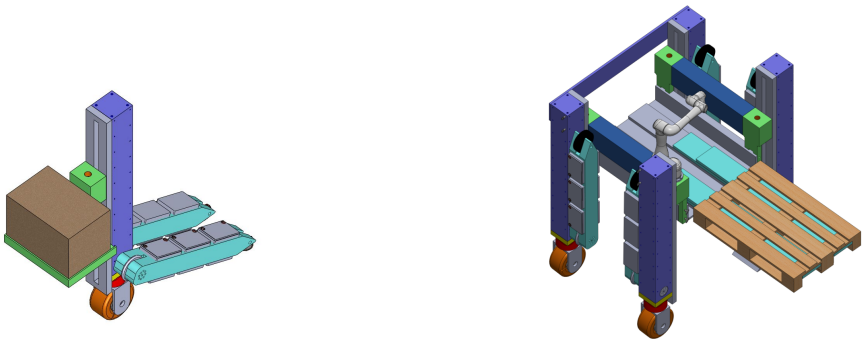


Figure 23: Reconfiguration from a 3-bot and a 2-bot into a 4-bot and a 1-bot (R_i denotes the i-th elementary robot)

Figure 24 shows two examples of M3-Cooper poly-robots of MecaBotiX carrying loads. The mono-bot in Figure 24a can carry a box while the quadri-bot in Figure 24b can carry a pallet.



(a) Mono-bot (1-bot) **(b) Quadri-bot (4-bot)**

Figure 24: Examples of reconfigurable poly-robots M3-Cooper [MecaBotiX, 2023]

The issue is to find out the optimal number of elementary robots and in which configuration, which type of load and in which period of time the loads should be transported to minimize the transportation cost or time.

1.7 Organization of the manuscript

The first chapter has presented the context of the thesis and given an overview of existing automated solutions in warehouses, in particular cooperative mobile robots and the state of the art of the fleet problems.

The second chapter treats the transport of identical loads by a fleet of non-cooperative and cooperative robots.

The third chapter deals with the sizing of a fleet of reconfigurable robots for the transport of heterogeneous loads. This chapter also compares the number of elementary robots in the cases with and without reconfiguration.

The fourth chapter studies the complexity of the problem and provides a heuristic algorithm with a numerical experiment.

The fifth chapter gives an extension of the problem, posing the question of makespan minimisation of a given fleet.

At the end we sum up the results and look at further ways of plot development.

Chapter 2

Transport of homogeneous loads

Contents

In this chapter, we are interested in the sizing of a fleet of non-cooperative and cooperative robots for the transport of standardized loads that are all identical and referred as "homogeneous loads". We consider the problem of determining the number of robots necessary to transport a set of homogeneous loads in a given time interval from a zone A to a zone B, at minimum cost. The cost is function of the number of robots and of the distance travelled by robots. The operations are divided into several phases: loading, loaded travel forth, unloading, empty travel back and battery charging.

To our knowledge, this chapter is the first scientific work to focus on the sizing of a fleet of cooperative robots. The results obtained in this chapter have been published in [Chaikovskaia et al., 2021].

2.1 Non-cooperative robots

In this first part, we consider the sizing of a fleet of non-cooperative robots and extend the results of Rjeb et al. [2021b] by adding the concept of transport capacity.

2.1.1 Assumptions and notations

The following notations are used:

- d: round trip distance from A to B to A
- τ: cycle time of the round trip, including loading/unloading time
- v_l: travelling speed of a loaded robot
- v_e: travelling speed of a empty robot
- t_l: loading time
- t_u: unloading time
- t_b: average immobilization duration for a single robot, due to battery recharge, maintenance, failure
- c: robot capacity
- D: total distance traveled by all the robots

- α: fixed cost per unit of time of a robot
- β: cost per meter traveled by a robot
- γ: fixed cost per unit of time, independent of the number of robots
- N: number of robots for loads transportation
- n: number of transported loads
- r: number of round trips for a robot
- T: planning horizon

We consider a fleet of N identical mobile robots which must transport a set of n identical loads from zone A to zone B, as shown in Figure 25.

Robot capacity is denoted c. The capacity of a robot is related to the size and mass of the loads and the number of loads that a robot can carry simultaneously.

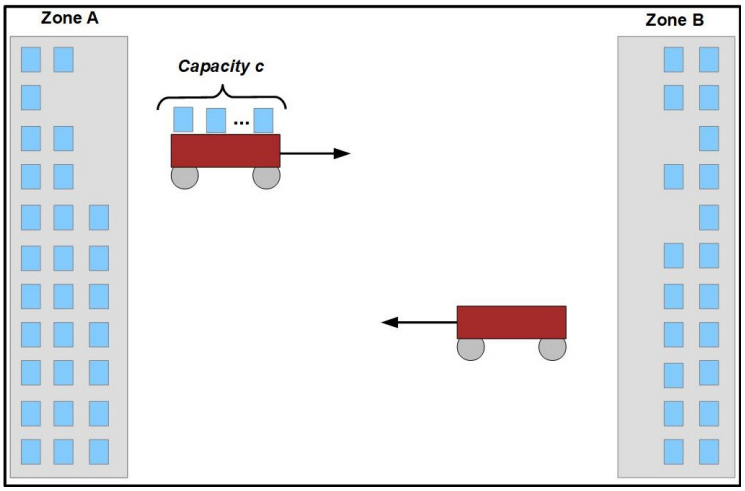


Figure 25: Transport of identical loads by a fleet of homogeneous robots

The cycle time τ represents the sum of the loaded travel time ($d/2v_l$ where v_l is the travelling speed of a loaded robot), the empty travel time ($d/2v_e$ where v_e is the displacement speed of an empty robot), the loading time (t_l)and the unloading time (t_u):

$$\tau = t_l + \frac{d}{2v_l} + t_u + \frac{d}{2v_e} \tag{2.1}$$

Over the time interval $[0, T]$, the robot is immobilized during t_b (battery recharge, maintenance, failure, etc.). The remaining available time is then $(T - t_b)$.

The cost per unit of time of a fleet of $N > 0$ robots traveling a total distance D over $[0, T]$ is

$$f(N) = \alpha N + \frac{\beta D}{T} + \gamma \tag{2.2}$$

where α represents the fixed cost of a robot per unit of time (cost related to maintenance, purchase or rental), β the cost per meter traveled by one robot and γ the cost per unit of time independent of the number of robots (e.g. hardware and software infrastructure).

Note that the total distance traveled D is directly related to the number of loads n to be transported. It takes $r = \frac{n}{c}$ round trips to transport the n loads. Then $D = dr$ and the cost function is

$$f(N, r) = \alpha N + \frac{\beta d}{T} r + \gamma. \tag{2.3}$$

The objective is to determine the number of required robots, N^*, to transport the set of n loads over time interval $[0, T]$ at minimum cost, considering A as the starting point of the robots. This simple problem is equivalent to determining the minimum number of round trips and robots allowing all loads to be transported over the time interval. To have feasible solutions, we assume that $T - t_b \geq \tau$.

The following assumptions are also made:

- The robot storage place is located at point A. There is no waiting to load in A or to unload in B (the loads are available immediately to be loaded and the robots do not hinder each other).
- The problem of traffic jams for robots is not taken into account. These different elements could nevertheless be taken into account by introducing an efficiency coefficient, as proposed in Egbelu [1987].

2.1.2 Minimal number of robots (finite horizon)

One robot can make at most $\left\lfloor \frac{T - t_b}{\tau} \right\rfloor$ round trips over the time interval $[0, T]$ where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Thus N robots with capacity c can carry at most $Nc \left\lfloor \frac{T - t_b}{\tau} \right\rfloor$ loads over the time interval.

To transport the n loads, it is therefore necessary that $Nc\left\lfloor \frac{T-t_b}{\tau} \right\rfloor \geq n$ and therefore that $N \geqslant \frac{n}{c\lfloor (T-t_b)/\tau \rfloor}$.

The number of robots being an integer, the minimum number of robots to transport n loads during the time interval $[0, T]$ is then

$$N^* = \left\lceil \frac{n}{c\lfloor (T-t_b)/\tau \rfloor} \right\rceil \tag{2.4}$$

The minimum number of round trips for robots to transport n loads is

$$r^* = \left\lceil \frac{n}{c} \right\rceil, \tag{2.5}$$

where $\lceil x \rceil$ is the least integer greater than or equal to x.

The minimum cost is then

$$f^* = f(N^*, r^*) = \alpha N^* + \frac{\beta d}{T} r^* + \gamma \tag{2.6}$$

$$= \alpha \left\lceil \frac{n}{c\lfloor (T-t_b)/\tau \rfloor} \right\rceil + \frac{\beta d}{T} \left\lceil \frac{n}{c} \right\rceil + \gamma \tag{2.7}$$

Consider the following example: $n = 5, c = 2, \tau = 0.4$, $t_b = 0, T = 1, \alpha = 10, \beta d = 2, \gamma = 1$. Then $N^* = 2$, $f^* = 27$ and a possible scheduling is represented as a Gantt diagram in Figure 26.

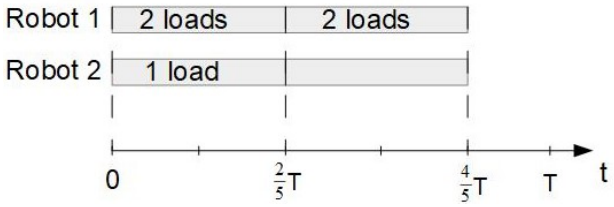


Figure 26: Gantt diagram for optimal transport of 5 loads

2.1.3 Minimal number of robots (infinite horizon)

We are also interested in the limit case where the time horizon T tends to infinity. This allows, on the one hand, to avoid side effects (if horizon T is not a multiple of cycle time τ) and, on the other hand, to model a fleet of vehicles operating permanently.

We denote by $\mu = c/\tau$ the maximum flow rate of loads per robot (maximum number of loads that a robot can carry per unit of time), by $\lambda = n/T$ the demand flow of loads to transport and by $\delta = t_b/T$ the immobilization rate.

Theorem 2.1.1. *If T tends to infinity, keeping λ and δ constant, then:*

$$N^* = \left\lceil \frac{\lambda}{\mu(1-\delta)} \right\rceil \tag{2.8}$$

$$f^* = \alpha \left\lceil \frac{\lambda}{\mu(1-\delta)} \right\rceil + \beta d \frac{\lambda}{c} + \gamma \tag{2.9}$$

Proof. Using the fact that $x - 1 < \lfloor x \rfloor \leq x$, we can limit the optimal number of robots obtained in (2.4):

$$\left\lceil \frac{n}{c\frac{T-t_b}{\tau}} \right\rceil \leq N^* < \left\lceil \frac{n}{c\left(\frac{T-t_b}{\tau} - 1\right)} \right\rceil \tag{2.10}$$

$$\Leftrightarrow \left\lceil \frac{\frac{n}{T}}{\frac{c}{\tau}\left(1 - \frac{t_b}{T}\right)} \right\rceil \leq N^* < \left\lceil \frac{\frac{n}{T}}{\frac{c}{\tau}\left(1 - \frac{t_b}{T}\right) - \frac{c}{T}} \right\rceil \tag{2.11}$$

Using the notations λ, μ, δ, this frame is re-written

$$\left\lceil \frac{\lambda}{\mu(1-\delta)} \right\rceil \leq N^* < \left\lceil \frac{\lambda}{\mu\left(1 - \delta - \frac{c}{T}\right)} \right\rceil \tag{2.12}$$

If we tend T to infinity, keeping λ and δ constant, we get

$$N^* \xrightarrow[T \to +\infty]{\frac{n}{T} \to \lambda,\ \frac{t_b}{T} \to \delta} \left\lceil \frac{\lambda}{\mu(1-\delta)} \right\rceil \tag{2.13}$$

Similarly, we have the framing

$$\frac{n}{cT} \leq \frac{1}{T}\left\lceil \frac{n}{c} \right\rceil < \frac{1}{T}\left(\frac{n}{c} + 1\right) \tag{2.14}$$

and

$$\frac{1}{T}\left\lceil \frac{n}{c} \right\rceil \xrightarrow[T \to +\infty]{\frac{n}{T} \to \lambda} \frac{\lambda}{c} \tag{2.15}$$

Then

$$f^* \xrightarrow[T \to +\infty]{\frac{n}{T} \to \lambda,\ \frac{t_b}{T} \to \delta} \alpha \left\lceil \frac{\lambda}{\mu(1-\delta)} \right\rceil + \beta d \frac{\lambda}{c} + \gamma \tag{2.16}$$

□

2.2 Cooperative robots

In this section, we assume that robots can cooperate to transport loads. We remind the concepts and terminology developed in Section 1.6:

- bot = An elementary robot, which is cannot be split into several robots.;
- p-bot = A set of p elementary robots, which cooperate on the same task.

2.2.1 Assumptions and notations

The following notations are used:

- p: number of bots constituting one p-bot
- c_1: 1-bot capacity
- c_p: p-bot capacity
- $c'_1 = c_p/p$: virtual capacity of a 1-bot that is part of a p-bot
- τ_1: 1-bot cycle time
- τ_p: p-bot cycle time (including possible cooperation time)
- α: fixed cost per unit of time of a bot
- β: cost per meter traveled by a bot
- γ: fixed cost per unit of time, independent of the number of bots
- N_1: number of 1-bots working alone
- N_p: number of p-bots
- n_1: number of loads transported by 1-bots working alone
- n_p: number of loads transported by p-bots
- T: planning horizon
- r_1: number of round trips for 1-bots

- r_p: number of round trips for p-bots
- n: number of loads to be transported from A to B

$$n = n_1 + n_p$$

- N: number of 1-bots in the fleet (including those working in a p-bot)

$$N = N_1 + pN_p$$

We will make the following additional assumptions:

- $\tau_1 \leq \tau_p$ as a p-bot may waste time in cooperation.
- There is no additional cost associated to a p-bot. The costs of a p-bot are simply those induced by the bots constituting it.
- There is no possible reconfiguration. A p-bot always remains a p-bot and a 1-bot always remains alone.

2.2.2 Optimal fleet (finite horizon)

The fleet sizing problem can then be modeled by the following mathematical program which aims at minimizing the cost function for cooperative robots $f_c(r_1, r_p, N_1, N_p)$:

$$f_c = \alpha(N_1 + pN_p) + \frac{\beta d}{T}\left(r_1 + pr_p\right) + \gamma \tag{2.17}$$

subject to:

$$n_1 \leq N_1 c_1 \left\lfloor \frac{T - t_b}{\tau_1} \right\rfloor \tag{2.18}$$

$$n_p \leq N_p c_p \left\lfloor \frac{T - t_b}{\tau_p} \right\rfloor \tag{2.19}$$

$$n = n_1 + n_p \tag{2.20}$$

$$\frac{n_1}{c_1} \leqslant r_1 \tag{2.21}$$

$$\frac{n_p}{c_p} \leqslant r_p \tag{2.22}$$

$$r_1, r_p, n_1, n_p, N_1, N_p \in \mathbb{N} \tag{2.23}$$

Constraints interpretation:

- Constraint (2.18): the number of loads carried by 1-bots must be less than or equal to the maximum number of loads that 1-bots can transport on interval $[0, T]$;
- Constraint (2.19): the number of loads carried by p-bots must be less than or equal to the maximum number of loads that p-bots can transport on interval $[0, T]$;
- Constraint (2.20): the sum of the number of loads carried by 1-bots and p-bots must be equal to the total number of loads to be transported;
- Constraint (2.21): the number of round trips for the transport of all the loads designated by 1-bots must be rounded up to allow the transport of all loads;
- Constraint (2.22): the number of round trips for the transport of all loads designated by p-bots must be rounded up to allow the transport of all loads.

When $\beta = 0$, the problem comes down to determining the minimum number of robots allowing all loads to be transported. When $\alpha = 0$, the problem comes down to achieving the optimal number of round trips to transport all the loads (a round trip from a p-bot counts as p round trips): $r_1^* = \left\lceil \frac{n_1^*}{c} \right\rceil$ and $r_p^* = \left\lceil \frac{n_p^*}{c} \right\rceil$, where n_1^* et n_p^* the optimal number of loads transported by 1-bots and p-bots, respectively, at which the cost is minimal.

We will distinguish three cases linked to the respective capacities of 1-bot and p-bot.

- $c_1 = 0$

  In this first case, we assume that a 1-bot can't carry a load on its own ($c_1 = 0$). This scenario may appear for a load of great mass, great volume or even great length. For example, as shown in Figure 27, the robot cannot transport a load much larger than itself for stability reasons.

(a) 1-bot (not stable) (b) p-bot with $p = 2$

Figure 27: Illustration for case $c_1 = 0$

  The problem then consists in determining the number of p-bots needed to carry all the loads and the results of Section 2.1 can be re-used. So we have:

$$N_p^* = \left\lceil \frac{n}{c_p \left\lfloor (T - t_b)/\tau_p \right\rfloor} \right\rceil, \qquad N_1^* = 0.$$

- $c_1 \geq c_1'$

  In this second scenario, a single 1-bot has a greater capacity than a 1 bot in p-bot configuration. Let us give a first example where this scenario occurs. If the capacity constraint is related to the transported mass and an additional pallet is needed in p-bot mode, then we lose mass capacity in p-bot mode. Another example would be the case where the load is transported by manipulator arms installed on the mobile platform in p-bot mode. Figure 28 shows the case when the robot loses its mass capacity due to the pallet, and thus the 1-bots transport more than the p-bots.

(a) p-bot with $p = 2$ (b) 2 1-bots

Figure 28: Illustration for case $c_1 \geq c_1'$

  We can easily show that it is optimal to use exclusively 1-bots. Assume that we use a p-bot for a round-trip in time interval $[t, t + \tau_p]$ to transport c_p loads. As $\tau_1 \leq \tau_p$, we could use p individual 1-bots to transport these loads in the same interval, as $c_1 \geq c_1'$ (or equivalently $pc_1 \geq c_p$). Hence, it is optimal to use exclusively 1-bots.

  We can then use again the results from the previous section. So we have

$$N_p^* = 0, \qquad N_1^* = \left\lceil \frac{n}{c_1 \left\lfloor (T - t_b)/\tau_1 \right\rfloor} \right\rceil.$$

- $0 < c_1 < c_1'$

  In this 3rd case, a single 1-bot has a lower capacity than a 1 bot in p-bot configuration. This scenario may arise for the transport of long objects (for example tubes) or even objects of large volumes but of low density. Figure 29 shows an example where a p-bot can have a capacity greater than a 1-bot, when we can not stack loads on top of each other.

(a) p-bot with $p = 2$ (b) 2 1-bots

Figure 29: Illustration for case $0 < c_1 < c'_1$

In this case, unlike previously, the optimal solution can consist of a mix of p-bots and 1-bots. This appears in particular if the cost linked to the traveled distance is significant. Consider the following example: $n = 4, p = 2, c_1 = 1, c_p = 3, \tau_1 = \tau_p = T/2, t_b = 0, \alpha = 1, \beta d/T = 10, \gamma = 0$. Table 2 presents the optimal solution according to the type of authorized robots.

	N_1	N_p	n_1	n_p	total cost
1-bot only	2	0	4	0	42
p-bot only	0	1	0	4	42
Mix of 1-bot and of p-bot	1	1	1	3	33

Table 2: Optimal solutions according to the types of authorized robots

Figure 30 shows the Gantt diagram of the optimal solution when both configurations are allowed.

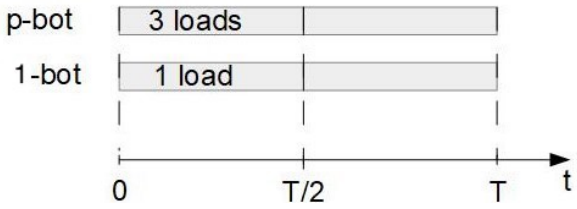


Figure 30: Gantt diagram of the optimal solution of 4 loads

2.2.3 Optimal fleet (infinite horizon)

We use the following additional notations:

- $\mu_1 = \dfrac{c_1}{\tau_1}$: 1-bot flow rate
- $\mu_p = \dfrac{c_p}{\tau_p}$: p-bot flow rate
- λ: demand flow rate of loads to be transported

- $\lambda_1 = \dfrac{n_1}{T}$: flow rate of loads carried by 1-bots alone
- $\lambda_p = \dfrac{n_p}{T}$: flow rate of loads carried by p-bots
- $\delta = t_b/T$ immobilization rate

The mathematical program can then be written as a MILP (Mixed-Integer Linear Programming) :

$$\min \quad \alpha(N_1 + pN_p) + \beta d\left(\frac{\lambda_1}{c_1} + p\frac{\lambda_p}{c_p}\right) + \gamma \tag{2.24}$$

$$s.t. \quad \lambda_1 \leq N_1\mu_1(1-\delta) \tag{2.25}$$

$$\lambda_p \leq N_p\mu_p(1-\delta) \tag{2.26}$$

$$\lambda = \lambda_1 + \lambda_p \tag{2.27}$$

$$\lambda_1, \lambda_p \in \mathbb{R}, \quad N_1, N_p \in \mathbb{N} \tag{2.28}$$

In two cases, we can re-use the results of Section 2.1.3.

$c_1 = 0$

$$N_1^* = 0, \qquad N_p^* = \left\lceil \frac{\lambda}{\mu_p(1-\delta)} \right\rceil.$$

$c_1 \geq c_1'$

$$N_1^* = \left\lceil \frac{\lambda}{\mu_1(1-\delta)} \right\rceil, \qquad N_p^* = 0.$$

Conclusion

The case of non-cooperative robots was considered for which we have derived a closed-form expression for the optimal number of robots. Then we have considered the case of cooperative robots where loads can be carried either by a single robot (1-bot) or by several robots that cooperate (p-bot). The fleet sizing problem can be formulated as a mathematical programming. We have distinguished several scenarios, depending on the respective carrying capacity of 1-bots and p-bots. Our mathematical model allows us to determine the most profitable number of robots that should cooperate. If the capacity of p 1-bots is smaller than the capacity of a single p-bot, then using exclusively p-bots or with a mix of 1-bots can lead to a significant cost decrease. Otherwise, it is optimal

to use exclusively 1-bots. We have also addressed the infinite horizon problem which models a fleet of vehicles operating permanently and leads to simpler results.

Chapter 3

Transport of heterogeneous loads

Contents

In this and later chapters, we consider elementary robots that can be connected in different ways in order to transport loads of different types. These robots can be assembled in different ways over time to adapt to the loads to be transported. This connection and disconnection between robots are called reconfiguration. The objective is to determine the optimal number of elementary robots required to transport various loads within a specified time frame. We formulate this problem as an integer linear program. Then, we investigate the special cases with two types of loads, two allowed configurations (1-bot and p-bot with $p \geq 2$) and unit capacities.

Some of the results obtained in this chapter have been published in [Chaikovskaia et al., 2022].

3.1 Assumptions and notations

We consider a fleet of N mobile elementary robots able to cooperate to transport loads of different types. An elementary robot is abbreviated as *bot*. A p-bot is a configuration where p elementary robots cooperating on the same transportation task. A 1-bot is an elementary robot working alone. A maximum of P elementary robots can cooperate. Therefore the set of possible configurations is $\{1, \cdots, P\}$.

There are n_k loads of type k to be transported ($k = 1, \cdots, K$). All the loads to be moved are located in the loading area of the warehouse and must be transported by the p-bots to the unloading area. A p-bot can only carry one type of load at a time and can simultaneously carry c_{pk} loads of type k. Hence, c_{pk} is called the capacity of the p-bot. We assume that for each type of load there is at least one configuration capable of carrying it.

The time horizon is divided into T periods ($t = 1, \cdots, T$). At the beginning of each period, the robots are located in the loading area and can be reconfigured. Note that the time of reconfiguration is not taken into account.

In each period, after the reconfiguration of robots, p-bots have time to complete the following four steps of a delivery, namely: loading, loaded trip, unloading and empty return trip.

We consider three types of costs: a fixed cost per elementary robot linked to its acqui-

sition or rental, a variable cost linked to the distance traveled by the robots and a fixed cost independent of the size of the fleet (hardware and software infrastructure). The fixed cost of an elementary robot is denoted α. The cost of a round trip for an elementary robot is denoted β and it follows that the cost of a round trip for a p-bot is $p\beta$. Finally, the fixed cost of a fleet is denoted γ. Thus, the cost of a fleet of N elementary robots performing M round trips is $\alpha N + \beta M + \gamma$. In the following, we won't consider γ because it is a constant in the optimization problem.

The objective is to determine the number of elementary robots needed to transport all the loads over the time horizon at minimum cost. We consider two variants. In the first variant, reconfiguration is prohibited and the configurations of the robots are fixed over the entire horizon. The optimal number of elementary robots is denoted by N_W in this case. In the second variant, reconfiguration is allowed and the configurations can be changed at the start of each period. The optimal number of elementary robots is denoted N_R in this case. If we do not consider the cost related to the travelled distance, i.e. if we take $\beta = 0$, then N_R and N_W represent the minimum number of robots with or without reconfiguration respectively.

We now remind the main notations introduced in this section:

- k: index for load type ($k \in \{1, \cdots, K\}$)
- p: index for configuration ($p \in \{1, \cdots, P\}$)
- t: index for time period ($t \in \{1, \cdots, T\}$)
- n_k: number of loads of type k
- c_{pk}: capacity of a p-bot carrying loads of type k
- α: fixed cost of the acquisition of an elementary robot
- β: cost of a round trip for an elementary robot
- N_W: optimal number of elementary robots when reconfiguration is prohibited
- N_R: optimal number of elementary robots when reconfiguration is allowed

3.2 Mathematical formulations

In this part, we propose two ILPs in order to formulate the two variants of the problem of minimizing the number of bots, with or without reconfiguration.

3.2.1 Without reconfiguration

We first assume that reconfiguration is prohibited. We use the following decision variables:

- N_p: total number of p-bots
- N_{pk}^t: number of p-bots carrying loads of type k in period t

The number of elementary robots is then $N_W = \sum_{p=1}^{P} pN_p$. The optimization problem can be formulated by the following ILP:

$$\min \quad \alpha \sum_{p=1}^{P} p \cdot N_p + \beta \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N_{pk}^t \tag{3.1}$$

subject to :

$$\sum_{t=1}^{T} \sum_{p=1}^{P} c_{pk} \cdot N_{pk}^t \geq n_k \qquad \forall k \tag{3.2}$$

$$N_p \geq \sum_{k=1}^{K} N_{pk}^t \qquad \forall t, \forall p \tag{3.3}$$

$$N_p \in \mathbb{N},\ N_{pk}^t \in \mathbb{N} \qquad \forall k, \forall p, \forall t \tag{3.4}$$

Constraint (3.2) means that the total capacity of the fleet along the time horizon must be able to transport all loads of each type. Constraint (3.3) means that the number N_p of p-bots must be greater than or equal to the number of p-bots used over each period.

3.2.2 With reconfiguration

When the reconfiguration is allowed, we use the same decision variables, with the difference that $N_R = N$ because each p-bot can be split into p number of bots, and the

problem can be formulated by the following ILP:

$$\min \quad \alpha N + \beta \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N_{pk}^{t} \tag{3.5}$$

subject to :

$$\sum_{t=1}^{T} \sum_{p=1}^{P} c_{pk} \cdot N_{pk}^{t} \geq n_k \qquad \forall k \tag{3.6}$$

$$N \geq \sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N_{pk}^{t} \qquad \forall t \tag{3.7}$$

$$N \in \mathbb{N},\ N_{pk}^{t} \in \mathbb{N} \qquad \forall k, \forall p, \forall t \tag{3.8}$$

Constraint (3.6) is similar to that of the problem without reconfiguration. Constraint (3.7) means that the number of elementary robots used, N, must be greater than or equal to the number of elementary robots used over each period.

3.3 Minimizing the number of robots

We assume in all this section that $\beta = 0$. Therefore N_R and N_W represent the minimum number of elementary robots needed to carry all the loads over the time horizon with or without reconfiguration respectively. In this section, we derive first closed-form formulas for the optimal number of robots for two types of loads. Then we study the case with unit capacities.

3.3.1 Two types of loads

We suppose that we have a wareshouse that handles only boxes and pallets. We obtain additional results for the special case of two types of loads and two possible configurations (1-bot or p-bot with $p \geq 2$). We assume that there are $n_1 \geq 1$ loads of type 1 and $n_2 \geq 1$ loads of type 2. Capacities of each configuration are summarized in Table 3. We can imagine that loads of type 1 and 2 correspond to respectively small (S) and medium (M) loads.

Configuration	Type of load	
	$k = 1$ (S)	$k = 2$ (M)
1-bot	c_{11}	0
p-bot	c_{p1}	c_{p2}

Table 3: Capacity matrix for 2 types of loads

A 1-bot can carry up to c_{11} loads of type 1 but no load of type 2. A p-bot can carry up to c_{p1} loads of type 1 and up to c_{p2} loads of type 2. In order to always have feasible solutions, we assume that $c_{11} \geq 1$ and $c_{p2} \geq 1$. When $c_{p1} = 0$, it corresponds to a dedicated transport: loads of type 1 (respectively type 2) can only be transported by 1-bots (respectively p-bots).

The problem simplifies considerably because there is only one way to transport loads of type 2 and we can obtain analytical formulas for N_R and N_W. Moreover, when $c_{p1} \geq p \cdot c_{11}$, reconfigurability does not allow to reduce the number of robots. Theorem 3.3.1 summarizes these results. In this theorem, $x^+ = \max(0, x)$ denotes the positive part of x.

Theorem 3.3.1 (Two types of loads). *Let* $n_1' = T \cdot \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil - \left\lceil \frac{n_2}{c_{p2}} \right\rceil$, $n_1^r = (n_1 - n_1' \cdot c_{p1})^+$ *and* $\tilde{n}_1^r = \left(n_1^r - \left\lfloor \frac{n_1^r}{T \cdot c_{p1}} \right\rfloor T \cdot c_{p1} \right)^+$.

If $c_{p1} < p \cdot c_{11}$, *then*

$$N_W = p \cdot \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + \left\lceil \frac{n_1^r}{T \cdot c_{11}} \right\rceil, \tag{3.9}$$

$$N_R = p \cdot \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + \left\lceil \frac{(n_1 - n_1' \cdot p \cdot c_{11})^+}{T \cdot c_{11}} \right\rceil. \tag{3.10}$$

If $c_{p1} \geq p \cdot c_{11}$, *then*

$$N_W = N_R = p \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + p \left\lfloor \frac{n_1^r}{T \cdot c_{p1}} \right\rfloor + \min \left(\left\lceil \frac{\tilde{n}_1^r}{T \cdot c_{11}} \right\rceil, p \right). \tag{3.11}$$

Proof. We distinguish two cases. Whatever the case, we assume that the p-bots are fully filled before a new p-bot is loaded so that we have no more than one p-bot with free periods at the outcome of the assignment of loads of type 2.

Case 1 : $c_{p1} < p \cdot c_{11}$ In this case, note that it is always more interesting to use 1-bots to transport loads of type 1. Let's start by determining N_W. The general idea is as follows:

1. We first assign loads of type 2 to p-bots;
2. If the last p-bot is not used over all the periods, we assign as many loads of type 1 as possible to the last p-bot assigned for loads of type 2;
3. The remaining loads of type 1 are assigned to 1-bots.

In step (1), the number of p-bots needed to carry loads of type 2 is

$$\left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil . \tag{3.12}$$

We assume that the p-bots are fully filled before a new p-bot is loaded, so that there is no more than one p-bot with free periods at the end of the assignment of loads the second type. Let n_1' be the number of periods not used by the last p-bot:

$$n_1' = \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil T - \left\lceil \frac{n_2}{c_{p2}} \right\rceil . \tag{3.13}$$

Indeed, it takes $\left\lceil \frac{n_2}{c_{p2}} \right\rceil$ periods to transport loads of type 2 with p-bots and the total available number of periods for p-bots is $\left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil T$.

In step (2), we can therefore assign up to $n_1' \cdot c_{p1}$ loads of type 1 to the last p-bot.

In step (3), it remains $n_1^r = (n_1 - n_1' \cdot c_{p1})^+$ loads of type 1 that requires $\left\lceil \frac{n_1^r}{T \cdot c_{11}} \right\rceil$ 1-bots.

We conclude that

$$N_W = p \cdot \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + \left\lceil \frac{n_1^r}{T \cdot c_{11}} \right\rceil . \tag{3.14}$$

We now determine N_R. There are three steps as for the calculation of N_W. Step (2) is modified as follows: if the last p-bot is not used over all the periods, it is reconfigured into p 1-bots to which we assign as many loads of type 1 as possible (at most $n_1' \cdot p \cdot c_{11}$). For step (3), then $(n_1 - n_1' \cdot p \cdot c_{11})^+$ loads of type 1 remain to be transported by 1-bots. We conclude that

$$N_R = p \cdot \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + \left\lceil \frac{(n_1 - n_1' \cdot p \cdot c_{11})^+}{T \cdot c_{11}} \right\rceil . \tag{3.15}$$

Case 2 : $c_{p1} \geq p \cdot c_{11}$

We assume that a p-bot has more capacity than p 1-bots to carry loads of type 1. It is then always more interesting to use a p-bot than p 1-bots. Thus, there is no point for reconfiguration and $N_W = N_R$.

We now compute N_W ($= N_R$). The general idea is as follows:

1. We first assign loads of type 2 to p-bots;
2. We assign as many loads of type 1 as possible to the last p-bot (assigned for loads of type 2), which may have free periods;
3. We assign the remaining loads of type 1, if any, to p-bots, as long as we can use them at maximum capacity on the whole horizon;
4. We assign the last loads of type 1, if any, to 1-bots if it requires less than p 1-bot and to a p-bot otherwise.

In step (1), the number of p-bots needed to carry loads of type 2 is

$$\left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil . \tag{3.16}$$

We assume that the p-bots are fully filled before a new p-bot is loaded, so that there is no more than one p-bot with free periods at the end of the assignment of loads the second type. Let n_1' be the number of periods not used by the last p-bot:

$$n_1' = \left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil T - \left\lceil \frac{n_2}{c_{p2}} \right\rceil . \tag{3.17}$$

Indeed, it takes $\left\lceil \frac{n_2}{c_{p2}} \right\rceil$ periods to transport loads of type 2 with p-bots and the total available number of periods for p-bots is $\left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil T$.

In step (2), we can assign $n_1' \cdot c_{p_1}$ loads of type 1 to the last p-bot. There remains then $n_1^r = (n_1 - n_1' \cdot c_{p1})^+$ loads of type 1 to transport.

In step (3), the additional number of p-bots required to transport loads of type 1 (using full capacity on the whole horizon) is

$$\left\lfloor \frac{n_1^r}{T \cdot c_{p1}} \right\rfloor . \tag{3.18}$$

At the end of step (3), it remains $\tilde{n}_1^r = \left(n_1^r - \left\lfloor \frac{n_1^r}{T \cdot c_{p1}} \right\rfloor T \cdot c_{p1} \right)^+$ loads of type 1 to transport.

In step (4), we assign the last $\tilde{n}_1^r$ loads of type 1 to 1-bots if it requires less than p 1-bot and to a p-bot otherwise. Hence we need the following additional number of elementary robots :

$$\min\left(\left\lceil \frac{\tilde{n}_1^r}{T \cdot c_{11}} \right\rceil, p\right). \tag{3.19}$$

In the end,

$$N_W = N_R = p\left\lceil \frac{n_2}{T \cdot c_{p2}} \right\rceil + p\left\lfloor \frac{n_1^r}{T \cdot c_{p1}} \right\rfloor + \min\left(\left\lceil \frac{\tilde{n}_1^r}{T \cdot c_{11}} \right\rceil, p\right). \tag{3.20}$$

□

3.3.2 Unit capacities

We now consider K types of loads and $P = K$ configurations with 0-1 capacities. More precisely, the capacity matrix is lower triangular and is such that c_{kp} is equal to 1 if $p \geq k$ and 0 otherwise. We also assume that there is at least one load of type K ($n_K \geq 1$), otherwise the problem reduces to a problem with $(K-1)$ types of loads.

Table 4 illustrates the capacity matrix with $K = 3$ types of loads. We can imagine that loads of type 1, 2, 3 correspond to respectively small (S), medium (M) and large (L) loads. Then small loads can be transported by all configurations, medium loads by 2-bots or 3-bots and large loads only by 3-bots.

Configuration	Type of load		
	$k = 1$ (S)	$k = 2$ (M)	$k = 3$ (L)
1-bot	1	0	0
2-bot	1	1	0
3-bot	1	1	1

Table 4: Capacity matrix for 3 types of loads

When considering such capacity structure, we are able to derive simple formulas for the minimum number of elementary robots with or without reconfigurations. In the case without reconfiguration, we obtain a result that holds for an arbitrary number of types of loads K. In the case with reconfiguration, we obtain a result up to 3 types of loads. With 4 types of loads or more, the problem becomes much more complex as the reconfiguration decision is more complex as we can, for instance, reconfigure a 4-bot into a 3-bot plus a 1-bot or into two 2-bots.

Theorem 3.3.2 (Triangular unit capacities)**.** *Let* $n'_K = 0$ *and for* $k = K, \cdots, 2$

$$N_k = \left\lceil \frac{(n_k - n'_k)^+}{T} \right\rceil \tag{3.21}$$

$$n'_{k-1} = n'_k - n_k + N_k \cdot T \tag{3.22}$$

Then

$$N_W = \sum_{k=1}^{K} k \cdot N_k \tag{3.23}$$

and, for $K \leq 3$,

$$N_R = \sum_{k=2}^{K} k \cdot N_k + \left\lceil \frac{(n_1 - n'_1 - \sum_{k=1}^{K-1} n'_k)^+}{T} \right\rceil \tag{3.24}$$

Note that (3.24) holds for $K > 3$ if we additionally assume that there are enough loads of type 1 to fill the holes ($n_1 \geq K(T-1)$). In this theorem, N_k represents the number of required k-bots and n'_k the number of free periods in the last used configuration after the assignment of loads of type $k+1, \cdots, K$.

Proof. Assume in all this proof that $K = P$ and that c_{kp} is equal to 1 if $p \geq k$ and 0 otherwise.

Without reconfiguration

We begin by exploring the case without reconfiguration. Note that, if you have a p-bot, it is optimal to use it as a priority to transport loads with a higher index k. Remind that we denote by N_k the minimum number of required k-bots. In what follows, we determine $N_k, N_{k-1}, \cdots, N_1$ in this order.

We also denote by n'_k the number of free periods in the last used configuration after the assignment of loads of type $k+1, \cdots, K$. It is optimal to use these free periods in priority for loads of type k, then of type $k-1$ and so on. Figure 31 illustrates notation of n'_k. In this figure, $n'_3 = 4$ means that we have 4 free periods for the transport of the loads of type 3 (L). $n'_2 = 2$ means that we have 2 free periods for the transport of loads of type 2 (M). $n'_1 = 3$ means that we have 3 free periods for the transport the loads of type 1 (S).

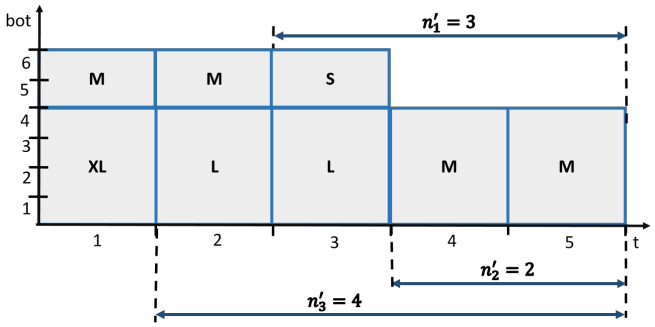


Figure 31: Illustration of notation n'_k with four types of loads
($T = 5, n_1 = 1, n_2 = 4, n_3 = 2, n_4 = 1$)

The number of K-bots necessary to transport loads of type K is

$$N_K = \left\lceil \frac{n_K}{T} \right\rceil. \tag{3.25}$$

It then may remain available capacity for the last K-bot. More precisely, the number of free periods for the last K-bot is

$$n'_{K-1} = N_K \cdot T - n_K = \left\lceil \frac{n_K}{T} \right\rceil T - n_K. \tag{3.26}$$

These n'_{K-1} free periods are used to transport in priority loads of type $(K-1)$ and it remains to transport $(n_{K-1} - n'_{K-1})^+$ loads of type $K-1$. The number of $(K-1)$-bots necessary to transport these remaining loads of type $(K-1)$ is then

$$N_{K-1} = \left\lceil \frac{(n_{K-1} - n'_{K-1})^+}{T} \right\rceil. \tag{3.27}$$

Assume now that the number of p-bots, N_p, has been determined for $p = k+1, \cdots, K$ and that there remains n'_k free periods on these configurations. These n'_k free periods are assigned in priority to loads of type k and then $(n_k - n'_k)^+$ loads of type k remain to be carried. The number of additional k-bots required is

$$N_k = \left\lceil \frac{(n_k - n'_k)^+}{T} \right\rceil. \tag{3.28}$$

The number of free periods for the last k-bot is $N_k \cdot T - (n_k - n'_k)^+$. There remains also $(n'_k - n_k)^+$ free periods after the transportation of loads of type j (for $j > k$). It follows that

$$n'_{k-1} = N_k \cdot T - (n_k - n'_k)^+ + (n'_k - n_k)^+ \tag{3.29}$$

$$= N_k \cdot T + n'_k - n_k. \tag{3.30}$$

In the end, we have

$$N_W = \sum_{k=1}^{K} k \cdot N_k \tag{3.31}$$

with

$$N_k = \left\lceil \frac{(n_k - n'_k)^+}{T} \right\rceil \tag{3.32}$$

$$n'_K = 0 \tag{3.33}$$

$$n'_{k-1} = n'_k - n_k + N_k \cdot T \qquad (\text{for } k = K, \cdots, 2) \tag{3.34}$$

With reconfiguration

$\underline{K = 1:}$ With a single type of load, we have immediately $N_R = \lceil \frac{n_1}{T} \rceil$ and it is easy to check that (3.24) gives the same result.

$\underline{K = 2:}$ When there are two types of loads, we need N_2 2-bots, as in the case without reconfiguration. If there remains free periods on the last 2-bot, it is optimal to reconfigure into two 1-bots. On the n'_1 free periods of the last 2-bot, we can transport up to $2n'_1$ loads of type 1. It remains $(n_1 - 2n'_1)^+$ loads of type 2 that require $\left\lceil \frac{(n_1 - 2n'_1)^+}{T} \right\rceil$ additional 1-bots. In the end, the optimal number of elementary robots is

$$N_R = 2 \cdot N_2 + \left\lceil \frac{(n_1 - 2n'_1)^+}{T} \right\rceil \tag{3.35}$$

and we have shown that (3.24) holds.

$\underline{K = 3:}$ Let's detail now the case with 3 types of loads. We need N_3 3-bots to transport loads of type 3, as in the case without reconfiguration. If there remains free periods on the last 3-bot, it is optimal to reconfigure into a 2-bot plus a 1-bot. On the n'_2 free periods of the last 3-bot, we can then transport up to n'_2 loads of type 2 and n'_2 loads of type 1. It remains $(n_2 - n'_2)^+$ loads of type 2 that require $N_2 = \left\lceil \frac{(n_2 - n'_2)^+}{T} \right\rceil$ additional 2-bots, as in the case without reconfiguration. On the last 2-bots, there are n'_1 free periods which can be used to transport up to $2n'_1$ loads of type 1. It remains $(n_1 - n'_2 - 2n'_1)$ loads of type 1 that require $\left\lceil \frac{(n_1 - n'_2 - 2n'_1)^+}{T} \right\rceil$ additional 1-bots. If ever $n_2 = 0$, then the 3-bot reconfigures

into three 1-bots and $n_2' = n_1'$. In the end, we have

$$N_R = 3 \cdot N_3 + 2 \cdot N_2 + \left\lceil \frac{(n_1 - n_2' - 2n_1')^+}{T} \right\rceil \tag{3.36}$$

and we have shown that (3.24) holds. □

3.4 Number of robots saved through reconfigurability

Again, we assume in all this section that $\beta = 0$ and N_R and N_W represent the minimum number of elementary robots (with or without reconfiguration respectively).

Below we show that we can at best divide the fleet size, by the number of types of loads K, using reconfiguration.

Theorem 3.4.1 (General case)**.** *For the problem described in Section 3.1, when $\beta = 0$, we have*

$$1 \leq \frac{N_W}{N_R} \leq K. \tag{3.37}$$

Proof. Let N_R be the minimum number of elementary robots with reconfiguration. N_R robots capable to transport not more than $\left(Tc_{p_k k} \left\lfloor \frac{N_R}{p_k} \right\rfloor\right)$ loads of type k, where $\forall k \quad p_k = \arg\max_p \left(c_{pk} \left\lfloor \frac{N_R}{p} \right\rfloor\right)$, to note that $\arg\max_x f(x)$ is the set of x for which $f(x)$ attains the function's largest value (if it exists). Then we have

$$n_k \leq T \left(c_{p_k k} \left\lfloor \frac{N_R}{p_k} \right\rfloor\right). \tag{3.38}$$

To transport n_k loads of type k, it suffices $\left(p_{min} \left\lceil \frac{n_k}{Tc_{p_{min}k}} \right\rceil\right)$ elementary robots, where $\forall k$ $p_{min} = \arg\min_p \left(p \left\lceil \frac{n_k}{Tc_{pk}} \right\rceil\right)$, to note that $\arg\min_x f(x)$ is the set of x for which $f(x)$ attains the function's smallest value (if it exists).

To transport all the loads without reconfiguration, it is necessary

$$N_W = \sum_k p_{min} \left\lceil \frac{n_k}{Tc_{p_{min}k}} \right\rceil \leq \sum_k \left(p_k \left\lceil \frac{n_k}{Tc_{p_k k}} \right\rceil\right) \leq \sum_k \left(p_k \left\lceil \frac{T\left(c_{p_k k} \left\lfloor \frac{N_R}{p_k} \right\rfloor\right)}{Tc_{p_k k}} \right\rceil\right) = \tag{3.39}$$

$$= \sum_k \left(p_k \left\lfloor \frac{N_R}{p_k} \right\rfloor\right) \leq \sum_k p_k \frac{N_R}{p_k} = KN_R \tag{3.40}$$

□

We now present an instance for which $\frac{N_W}{N_R}$ goes to K when P goes to infinity. Assume that c_{pk} equal 1 if $p = P - (K - k)$ and 0 otherwise.

Assume also that $P > K$ and that there is one load of each type. We have $N_R = P$ and $N_W = P + (P-1) + \cdots + (P - (K-1)) = KP - (1 + 2 + \cdots + (K-1)) = KP - \frac{K(K-1)}{2}$. Thus $\frac{N_W}{N_R} = K - \frac{K(K-1)}{2P}$ which goes to K when P goes to infinity.

3.4.1 Two types of loads

Once we showed that by allowing reconfiguration, the minimum number of robots can be divided by a factor up to K in general case, we now study the case of two types of loads described in Section 3.3.1. We can show that the number of robots saved by reconfigurability is at most p elementary robots.

Theorem 3.4.2 (Two types of loads)**.** *With two types of loads, the absolute gain is bounded as follows :*

$$0 \leq N_W - N_R \leq p. \tag{3.41}$$

Proof. When $c_{p1} \geq p \cdot c_{11}$, the result is trivial because $N_W = N_R$.

In the following, we assume that $c_{p1} < p \cdot c_{11}$. According to Theorem 3.3.1, we have

$$N_W - N_R = \left\lceil \frac{(n_1 - n_1' \cdot c_{p1})^+}{T \cdot c_{11}} \right\rceil - \left\lceil \frac{(n_1 - n_1' \cdot p \cdot c_{11})^+}{T \cdot c_{11}} \right\rceil \tag{3.42}$$

$$\leq \frac{(n_1 - n_1' \cdot c_{p1})^+}{T \cdot c_{11}} + 1 - \frac{(n_1 - n_1' \cdot p \cdot c_{11})^+}{T \cdot c_{11}} \tag{3.43}$$

$$= \frac{(n_1 - n_1' \cdot c_{p1})^+ - (n_1 - n_1' \cdot p \cdot c_{11})^+}{T \cdot c_{11}} + 1. \tag{3.44}$$

Three cases are then possible. Assume first that $n_1 \leq n_1' \cdot c_{p1}$. Then $(n_1 - n_1' \cdot c_{p1})^+ = (n_1 - n_1' \cdot p \cdot c_{11})^+ = 0$. With (3.44), it comes $N_W - N_R \leq 1 \leq p$.

Assume now that $n_1' \cdot c_{p1} < n_1 \leq n_1' \cdot p \cdot c_{11}$. Then we have $(n_1 - n_1' \cdot c_{p1})^+ = n_1 - n_1' \cdot c_{p1}$,

$(n_1 - n_1' \cdot p \cdot c_{11})^+ = 0$ and

$$N_W - N_R \leq \frac{n_1 - n_1' \cdot c_{p1}}{T \cdot c_{11}} + 1 \tag{3.45}$$

$$< \frac{n_1}{T \cdot c_{11}} + 1 \tag{3.46}$$

$$< \frac{n_1}{n_1' \cdot c_{11}} + 1 \tag{3.47}$$

$$\leq p + 1. \tag{3.48}$$

Strict inequality (3.46) comes from the assumption that $n_1' \cdot c_{p1} < n_1$. Strict inequality (3.47) comes from $n_1' < T$. Inequality (3.48) comes from the asssumption that $n_1 \leq n_1' \cdot p \cdot c_{11}$. We have $N_W - N_R < p + 1$ and, as $N_W - N_R$ and p are integers, we conclude that $N_W - N_R \leq p$.

Finally, assume that $n_1 \geq n_1' \cdot p \cdot c_{11}$. We can therefore remove the positive parts in (3.44):

$$N_W - N_R \leq \frac{n_1 - n_1' \cdot c_{p1}}{T \cdot c_{11}} + 1 - \frac{n_1 - n_1' \cdot p \cdot c_{11}}{T \cdot c_{11}} \tag{3.49}$$

$$= \frac{n_1'(p \cdot c_{11} - c_{p1})}{T \cdot c_{11}} + 1 \tag{3.50}$$

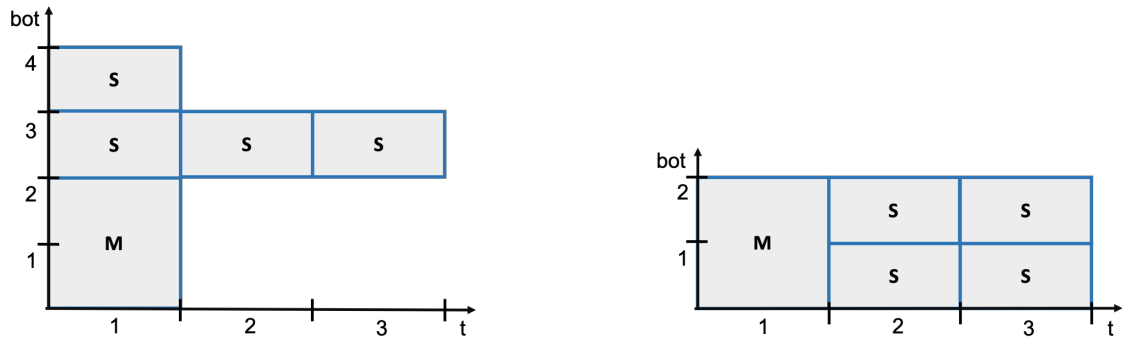
$$< \frac{p \cdot c_{11} - c_{p1}}{c_{11}} + 1 \tag{3.51}$$

$$\leq p + 1. \tag{3.52}$$

As $n_1' < T$, again $N_W - N_R < p + 1$ implies that $N_W - N_R \leq p$. In any case, we have $N_W \leq N_R + p$.

□

We now provide a simple example where the upper bounds of theorems 3.4.1 and 3.4.2 are reached. Let's take $T = 3$, $p = 2$, $c_{11} = c_{22} = 1$, $c_{12} = c_{21} = 0$, $n_1 = 4$ and $n_2 = 1$. Then $N_R = 2$ and $N_W = 4$. Loads of type 1, 2 correspond to respectively small (S) and medium (M) loads. Figure 32 represents the Gantt chart for this example. On the ordinate axis is the reference number of the bots, on the abscissa axis is the reference number of the period. In each rectangle is indicated the type of load that is transported. In the optimal strategy with reconfiguration, a 2-bot carries one medium load in period 1 and then is reconfigured into two 1-bots that carry the small loads in periods 2 and 3. In the optimal strategy without reconfiguration, the 2-bot can not be split and we need two additional 1-bots.



(a) Without reconfiguration

(b) With reconfiguration

Figure 32: Gantt chart for a simple example where the number of robots is halved when reconfiguration is allowed

We have seen that the gain can be important in the previous example with a fleet divided by two. However, the gain in relative value is limited for large robot fleets. Indeed, as a consequence of Theorem 3.4.2, we have

$$\frac{N_W}{N_R} \leq 1 + \frac{p}{N_R}. \tag{3.53}$$

For instance, if $N_R = 100$ and $p = 4$, then $N_W/N_R \leq 1.04$ and $N_W \leq 104$. Hence reconfigurability allows to divide the size of the fleet by at most a factor 1.04.

3.4.2 Unit capacities

We now compare the optimal numbers of robots with unit capacities, in the setting described in Section 3.3.2. As formula (3.24) holds only for $K \leq 3$, we assume that there are at most 3 types of loads.

Theorem 3.4.3 (Unit capacities)**.** *When $K \leq 3$, we have*

$$0 \leq N_W - N_R \leq K - 1. \tag{3.54}$$

Proof. From (3.23) and (3.24), we obtain that

$$N_W - N_R = \left\lceil \frac{(n_1 - n_1')^+}{T} \right\rceil - \left\lceil \left(\frac{n_1 - n_1'}{T} - \frac{1}{T} \cdot \sum_{k=1}^{K-1} n_k' \right)^+ \right\rceil. \tag{3.55}$$

Note that the number of free periods n'_k can not exceed $T-1$ as there is at least one period used to transport some load. Thus $n'_k \leq T-1$ for $k = 1, \cdots, K-1$. It follows that

$$N_W - N_R \leq \left\lceil \frac{(n_1 - n'_1)^+}{T} \right\rceil - \left\lceil \left(\frac{n_1 - n'_1}{T} - (K-1) \right)^+ \right\rceil. \tag{3.56}$$

To conclude we need the following property. Let b a non-negative integer and x a real number. Then $\lceil x^+ \rceil - \lceil (x-b)^+ \rceil \leq b$. The proof of this property is straightforward.

$$\begin{aligned} \lceil x^+ \rceil - \lceil (x-b)^+ \rceil &= \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq b \\ b & \text{if } x > b \end{cases} \\ &\leq b \end{aligned}$$

Using this property and (3.56) gives $N_W - N_R \leq K-1$. □

This theorem shows that the gain is not substantial in absolute value. When $K = 1$, the gain is null as expected. When $K = 2$, we can gain at most one elementary robot by using reconfigurability. When $K = 3$, the gain is at most of 2 elementary robots. Note that Theorem 3.4.3 holds for $K > 3$ if we additionally assume that there are enough loads of type 1 to fill the holes ($n_1 \geq K(T-1)$).

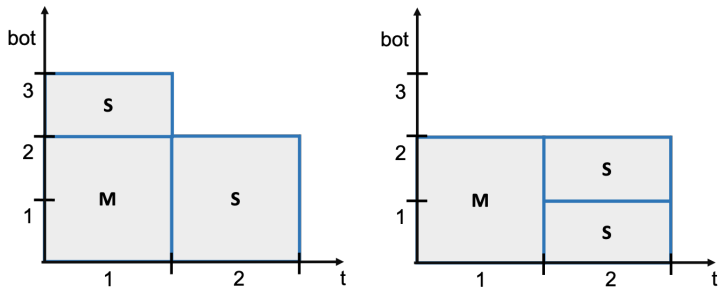
Theorem 3.4.3 implies that $\frac{N_W}{N_R} \leq 1 - \frac{K-1}{N_R}$. As $n_K \geq 1$, we have $N_R \geq K$ and we get that

$$1 \leq \frac{N_W}{N_R} \leq 2 - \frac{1}{K}. \tag{3.57}$$

In what follows, we provide examples where the upper bound in (3.54) and (3.57) are reached for two or three types of loads.

Example with two types of loads

In the following example, loads of type 1, 2 correspond to respectively small (S) and medium (M) loads. Let's take $T = 2$, $n_1 = 2$ and $n_2 = 2$. Then $N_R = 2$ and $N_W = 3$ and $\frac{N_W}{N_R} = \frac{3}{2}$. Figure 33 represents the Gantt chart for this example. On the ordinate axis is the reference number of the bots, on the abscissa axis is the reference number of the period. The type of loads the robot is carrying is indicated in each rectangle.



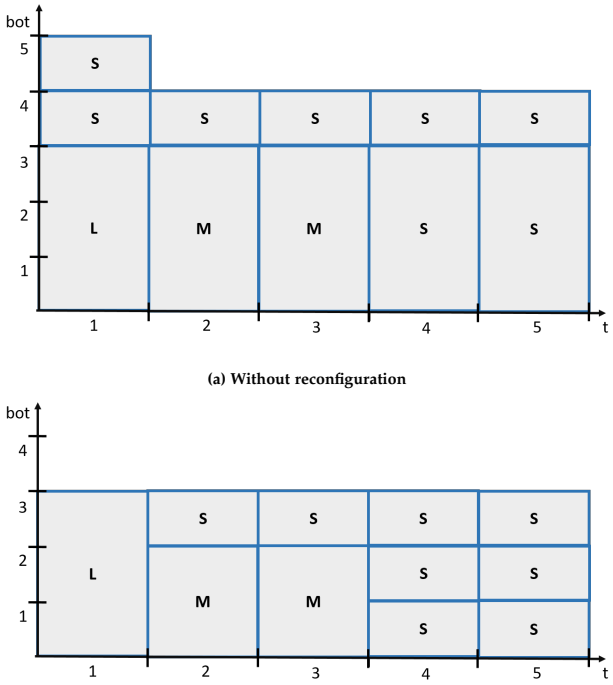
(a) Without reconfiguration (b) With reconfiguration

Figure 33: Gantt chart for an example with two types of loads

When reconfiguration is allowed, a 2-bot can carry one medium load, then it reconfigures into two 1-bots to carry 2 small loads. If the reconfiguration is not allowed, the 2-bot is not able to split into two independent elementary robots, so it carries a single small load.

Example with three types of loads

In the following example, loads of type 1, 2, 3 correspond to respectively small (S), medium (M) and large (L) loads. Let's take $T = 5$, $n_1 = 8$, $n_2 = 2$ and $n_3 = 1$. Then $N_R = 3$, $N_W = 5$ and $\frac{N_W}{N_R} = \frac{5}{3}$. Figure 34 represents the Gantt chart for this example.



(a) Without reconfiguration

(b) With reconfiguration

Figure 34: Gantt chart for an example with three types of loads

When reconfiguration is allowed, a 3-bot can carry one load of type L, then it reconfigures into 1-bot and 2-bot and then into three 1-bots to carry 6 small loads. If the reconfiguration is not allowed, the 3-bot is not able to split into two independent robots, so it carries a single large load and then two medium loads and two small loads.

3.4.3 Demand per period

For now, we have shown that the gain in relative value, N_W/N_R, can be significant while the gain in absolute value, $N_W - N_R$, remains limited. We will now propose a variant of the problem where the gain in absolute value can be significant.

We now assume that the demand to transport loads is per period and not over the whole horizon. More precisely, there are n_{kt} loads of type k to be transported in period t (instead of n_k loads of type k to be transported over the whole horizon in the original model). This may correspond, for example, to the activity of a warehouse that delivers one type of load on Monday and another type of load on Tuesday. With this new assumption, the ILPs remain unchanged except that constraints (3.2) and (3.6) have to be replaced by the following constraint:

$$\sum_{p=1}^{P} c_{pk} \cdot N_{pk}^{t} \geq n_{kt} \qquad \forall k, \forall t \tag{3.58}$$

We will now present a simple example where the gain in absolute value can be significant. Consider two types of loads and two periods where n_1 loads have to be transported in period 1 and n_2 loads have to be transported in period 2 (see Table 5b for a summary of demands). Type 1 (S) loads can only be transported by 1-bots while type 2 (M) loads can only be transported by p-bots with $p \geq 2$ (see Table 5a for a summary of capacities).

Configuration	Type of load	
	$k = 1$ (S)	$k = 2$ (M)
1-bot	1	0
p-bot	0	1

(a) Capacities

Type of load	Period	
	$t = 1$	$t = 2$
$k = 1$ (S)	n_1	0
$k = 2$ (M)	0	n_2

(b) Demands

Table 5: Example where the gain in absolute value can be significant

For this example, the minimum numbers of elementary robots are easy to derive :

$$N_W = n_1 + pn_2, \tag{3.59}$$

$$N_R = (n_1 - pn_2)^+ + pn_2 \tag{3.60}$$

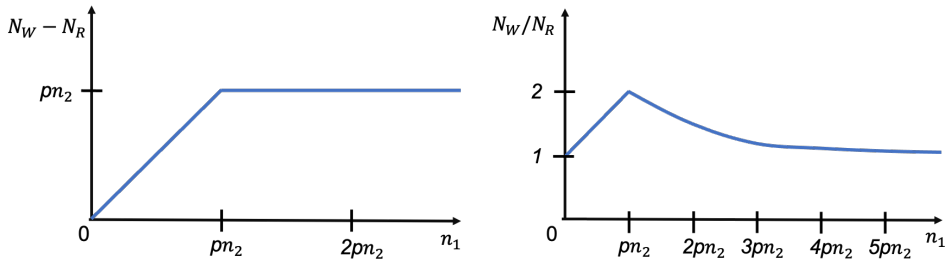
$$= \max(n_1, pn_2).$$

It follows that

$$N_W - N_R = \min(n_1, pn_2), \tag{3.61}$$

$$\frac{N_W}{N_R} = \begin{cases} 1 + \dfrac{n_1}{pn_2} & \text{if } n_1 \leq pn_2 \\ 1 + \dfrac{pn_2}{n_1} & \text{if } n_1 > pn_2. \end{cases} \tag{3.62}$$

Figure 35 plots the effect of n_1 on the difference $N_W - N_R$ and the ratio $\frac{N_W}{N_R}$. When $n_1 = pn_2$, the ratio N_W/N_R is maximum and equal to 2 while the difference is equal to n_1 and thus unbounded.



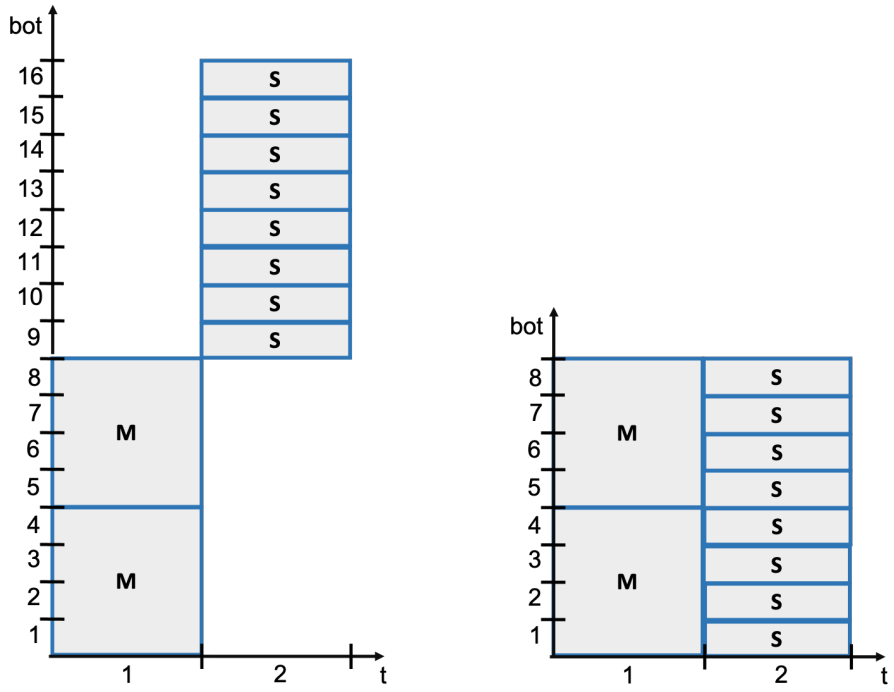
(a) Difference (b) Ratio

Figure 35: Effect of the number of loads of type 1 on the gains in absolute and relative value

Figure 36 represents the Gantt chart for the above example with $p = 4$, $n_1 = 8$ and $n_2 = 2$. We have then $N_R = 8$ and $N_W = 16$, $N_W - N_R = 8$ and $N_W/N_R = 2$.

Conclusion

In this chapter, mathematical programs were obtained in the form of ILP for load transportation with and without the possibility of reconfiguration. This allows us to find out which configuration of the robot in which period and for the transportation of which



(a) Without reconfiguration

(b) With reconfiguration

Figure 36: Gantt chart when $p = 4$, $n_1 = 8$ and $n_2 = 2$

type of load we will use to minimize the cost of the transportation. Thanks to the programs, we can compare the cost of the fleet with and without reconfiguration. These programs have been tested on Python and CPLEX Optimizer. For the special case of two types of loads and two configurations of robots, closed-form expressions are derived for the minimum number of robots with or without reconfiguration. A second special case with unit capacities is also studied. For this case, we have derived closed-form expressions for the minimum number of robots up to three load types with reconfiguration, and for any number of load types without reconfiguration. We have showed that reconfigurability can divide the minimum number of elementary robots up to a factor of K (with K the number of load types). For the two special cases, we show that the gain in number of robots is limited but may be significant for small fleets. Finally, in a variant where the demand is per period of time and not on the whole time horizon, we have showed that the gain in number of robots is not bounded.

Chapter 4

Complexity analysis and heuristic algorithms

Contents

In this chapter, we study the complexity of the problem, introduced in Chapter 3, where the transport of loads from one area to another is performed by reconfigurable mobile robots. After studying the complexity of the problem, we focus in this chapter on finding an effective heuristic to solve the problem in polynomial time.

4.1 Problem description and model formulation

4.1.1 The Multi_Bot problem

We consider here a fleet of mobile elementary robots which cooperate in order to transport loads of different types. An elementary robot is abbreviated as a *bot*. A *p-bot* is a configuration which makes p elementary bots cooperate on the same transportation task. A maximum of P elementary bots may cooperate, which identifies the feasible *configuration* set with the set **P** = {1,..., P}, P being one of the inputs of our problem.

There are K *load types* and we denote by $\mathbf{K} = \{1, \ldots, K\}$ the set of load types. The transportation demand for every type k is denoted by n_k. So K and vector **n** = (n_k, $k = 1, \ldots, K$) is another part of our inputs. Given a load type k, there is at least one value p such that related p-bot is able to transport load type k.

All transportation tasks must be executed within a discrete time horizon **T** = {1,..., T}. At the beginning of a given *period* t, the bots are located inside a loading area and may be put together (reconfigured) in order to provide us with the best fitted p-bots for the current period. During a given period t, a p-bot reconfigured this way may only deal with a single type k, and the number of loads of type k that it can transport cannot exceed a threshold c_{pk}. So T, as well as vector $C = (c_{pk}, p = 1, \ldots, P, k = 1, \ldots, K)$ becomes also part of our inputs.

Our purpose is to simultaneously minimize the number of bots involved into the process, and the number of *transportation transactions* (*trips*) performed throughout the periods $1, \ldots, T$. We see that if H_t denotes the number of bots required to be active during period t, then:

- The number of bots necessary to achieve the whole process is $H^{Max} = \max_t H_t$;
- The number of trips necessary to achieve the whole process is $H = \sum_t H_t$.

We consider two scaling coefficients α and β and our objective is to minimize the weighted cost:

$$Cost = \alpha \cdot H^{Max} + \beta \cdot H. \tag{4.1}$$

We call **Multi_Bot** problem the decisional problem described this way.

4.1.2 A Multi_Bot model

For any period t, configuration p and load type k, we denote by X^t_{pk} the number of trips which will take place at period t, and involve the transportation of load type k by a p-bot. Then we see that any H_t may be written $H_t = \sum_{k,p} p \cdot X^t_{pk}$, while the demand constraint related to the demand vector $n = (n_k, k = 1, \ldots, K)$ may be formulated as follows: For any load type k, $\sum_{t,p} c_{pk} \cdot X^t_{pk} \geq n_k$. It comes that our **Multi_Bot** problem may be formalized as follows.

Inputs:

- A set $\mathbf{P} = \{1, \ldots, P\}$ of configurations;
- A set $\mathbf{T} = \{1, \ldots, T\}$ of periods;
- A set $\mathbf{K} = \{1, \ldots, K\}$ of load types;
- A non negative integral vector $C = (c_{pk}, p = 1, \ldots, P, k = 1, \ldots, K)$, which represents capacities (the number of load types k which may be transported by a single p-bot during a period);
- A non negative integral vector $n = (n_k, t = 1, \ldots, K)$, which represents transportation demands.
- Two scaling non negative rational coefficient α and β.

Objective: Compute a non negative integral vector $X = (X^t_{pk}, t = 1, \ldots, T, p = 1, \ldots, P, k = 1, \ldots, K)$ such that:

- For any k, $\sum_{t,p} c_{pk} \cdot X^t_{pk} \geq n_k$;
- $\alpha \cdot (\max_t \sum_{k,p} p \cdot X^t_{pk}) + \beta \cdot (\sum_{t,p,k} p \cdot X^t_{pk})$ is the smallest possible.

Observe that the above formulation is not an ILP one. Still, it is easy to turn it into such an ILP model by introducing an additional variable Y that replaces $\max_t \sum_{k,p} p \cdot X_{pk}^t$ and adding the following constraints: for any t, $Y \geq \sum_{k,p} p \cdot X_{pk}^t$.

4.2 Structural properties of the Multi_Bot problem

Dealing with the design of algorithms, both for the specific cases when one among the parameters P, T, K is equal to 1 and for the general case, is going to require some better understanding of our **Multi_Bot** problem. The present section is going to provide us with additional information about **Multi_Bot** solutions.

4.2.1 Encoding size and decisional reformulation

Given a **Multi_Bot** instance *Multi_Bot*. **P**, **T**, **K** being defined as respectively configuration, period and load type sets, the encoding size of a *Multi_Bot* is:
$Size(Multi\_Bot) = P + T + K + \sum_{k,p}(1 + \lceil \log_2 c_{pk} \rceil) + \sum_k (1 + \lceil \log_2 n_k \rceil) + (1 + \log_2 \alpha) + (1 + \log_2 \beta)$.

As it is the case when a problem involves both combinatorial inputs and numbers [Garey and Johnson, 1979], this expression clearly distinguishes between the *combinatorial* part of the encoding size which is due to the sets **P**, **T**, **K** and the *numerical* part of this size related to numbers $\alpha, \beta, c_{p,k}$ and $n_k, k = 1, \ldots, K, p = 1, \ldots, P$.

We get a feasible **Multi_Bot** solution by setting $X_{pk}^t = n_k$ for any t, p, k. Related H value is equal to $K \cdot P \cdot T \cdot n^{Max}$, where $n^{Max} = \sup_k n_k$ and so the optimal cost value bounded by $2n^{Max} \cdot \sup(\alpha, \beta) \cdot K \cdot P \cdot T$. It comes that we may solve **Multi_Bot** by successively solving (binary search process) no more than $\lceil \log_2(2n^{Max} \cdot \sup(\alpha, \beta) \cdot K \cdot P \cdot T) \rceil$ instances of the following decision problem **Multi_Bot_dec**(S) where S is a threshold parameter bounded by $2n^{Max} \cdot \sup(\alpha, \beta) \cdot K \cdot P \cdot T$.

Multi_Bot_dec(S): Compute non negative integral vector $X = (X_{pk}^t, k = 1, \ldots, K, p = 1, \ldots, P, t = 1, \ldots, T)$ such that:

$$\sum_{t,p} c_{pk} \cdot X^t_{pk} \geq n_k \qquad \forall k \tag{4.2}$$

$$\alpha \cdot (\max_t \sum_{k,p} p \cdot X^t_{pk}) + \sum_{t,p,k} p \cdot X^t_{pk} \leq S \tag{4.3}$$

Since $\lceil \log_2(2n^{Max} \cdot \sup(\alpha, \beta) \cdot K \cdot P \cdot T) \rceil$ may be bounded by a polynomial function of *Size*(*Multi_Bot*), we see that the time-complexity of **Multi_Bot** is going to be (in terms of polynomiality or NP-Hardness) the same as the time-complexity (in terms of polynomiality or NP-Completeness) of **Multi_Bot_dec**(S).

4.2.2 Reinforcement of the Multi_Bot model

We are now going to see that we may impose additional constraints to the **Multi_Bot** model without deteriorating its optimal value. Those constraints will be the key for our algorithms.

For every k, let us set $p_0(k) = \arg\sup_p(c_{pk}/p)$. This quantity $p_0(k)$ is the most efficient configuration for the transportation of k.

Then the following lemma bounds the values $X^t_{pk}, p \neq p_0(k)$:

Lemma 4.2.1. *Given a **Multi_Bot** instance Multi_Bot. Imposing the following constraint* (4.4) *does not modify the optimal value of the instance Multi_Bot :*

$$X^t_{pk} \leq \frac{p_0(k)}{GCD(p_0(k), p)} - 1 \qquad \forall t, k, p \neq p_0(k) \tag{4.4}$$

In this lemma, GCD means Greatest Common Divisor.

Proof. Let X a feasible solution of the instance *Multi_Bot* and let us suppose that, for some (t, k) and for some $p_1 \neq p_0(k)$ we may write: $p_0(k) = u \cdot GCD(p_0(k), p_1); p_1 = v \cdot GCD(p_0(k), p_1); X^t_{p_1,k} \geq u$. Then we increase $X^t_{p_0(k),k}$ by v and decrease $X^t_{p_1,k}$ by u. Doing this maintains (4.3) since $u \cdot p_1 = v \cdot p_0$. But the inequality $p_0(k) \cdot c_{p_1,k} \leq p_1 \cdot c_{p_0(k),k}$ also keeps the quantity $\sum_{l,p} c_{pk} X^l_{pk}$ from decreasing and so (4.2) keeps holding. We can do this until (4.4) becomes satisfied and so we conclude. □

A consequence of above result is that we should focus, when dealing with any **Multi_Bot** instance, on the "critical" variables $X^t_{p_0(k),k}$. The algorithm that we are going to describe in Section 4.4 for the general case will derive from this interpretation of Lemma 4.2.1 and from the way we are going to characterize the complexity of **Multi_Bot**. It will also rely on a time polynomial pre-process (Section 4.3.2) which will allow us to store in advance the decisions related to variables $X^t_{p(k),k}$, $p \neq p_0(k)$.

Let us now set:

- For any t, $E_t = \sum_k p_0(k) \cdot X^t_{p_0(k),k}$ and $F_t = \sum_{p \neq p_0(k),k} p \cdot X^t_{pk}$;
- For any k, $Q(k) = (p_0(k) - 1) \cdot (P \cdot (P+1)/2 - p_0(k)) + p_0(k)$;
- $Q = \sum_k Q(k)$.

Q is bounded by a polynomial function of *Size*(*Multi_Bot*). Then the following lemmas 4.2.2 and 4.2.3 allow us to bound differences $|E_t - E_l|$ and $p_0(k) \cdot |X^t_{p_0(k),k} - X^l_{p_0(k),k}|$ in a way which will open the way to time-polynomial dynamic programming algorithms for the target cases when one among parameters P, T and K is equal to 1.

Lemma 4.2.2. *Given a **Multi_Bot** instance Multi_Bot. Then imposing the following constraint* (4.5) *does not modify the optimal value of the instance Multi_Bot :*

$$|E_t - E_l| = \left| \sum_k p_0(k) \cdot X^t_{p_0(k),k} - \sum_k p_0(k) X^l_{p_0(k),k} \right| \leq Q \qquad \forall t, l, t \neq l \qquad (4.5)$$

Proof. Let X a feasible solution of the instance *Multi_Bot*, which satisfies (4.4), and let us suppose that, for some t, l, we have $E_t - E_l > Q$. We may choose t_0 and l_0 with this property and such that E_{t_0} is maximal and E_{l_0} is minimal. Then must exist k_0 such $X^{t_0}_{p_0(k_0),k_0} \geq 1$, and so we may try to decrease $X^{t_0}_{p_0(k_0),k_0}$ by 1 and to increase $X^{l_0}_{p_0(k_0),k_0}$ by 1. Doing this keeps constraints (4.2) and (4.4). Also, we get $H_{t_0} \geq E_{t_0}$ and $H_{l_0} = E_{l_0} + F_{l_0} \leq E_{l_0} + \sum_{k,p \neq p_0(k_0)} (p_0(k_0) - 1) \cdot p \leq E_{l_0} + Q - p_0(k_0)$. It comes that even after that we increased $X^{l_0}_{p_0(k_0),k_0}$ by 1, the new value H_{l_0} does not exceed the previous value H_{t_0} and so H^{Max} did not increase. Neither did $H = \sum_t H_t$ so $\alpha H^{Max} + \beta H$ did not increase. But we also see that one of the following properties hold:

- The value $\max_t E_t - \min_t E_t$ decreased;

- The number of elements t which achieve $\max_t E_t$ or $\min_t E_t$ decreased.

It comes that applying above correction process as many times as necessary leads us to (4.5). □

Lemma 4.2.3. *Given a **Multi_Bot** instance Multi_Bot, reinforced by (4.4), (4.5). Then imposing the following constraint (4.6) does not modify the optimal value of the instance Multi_Bot:*

$$p_0(k) \cdot |X^t_{p_0(k),k} - X^l_{p_0(k),k}| \leq Q + TP^2 \qquad \forall t, l, t \neq l \tag{4.6}$$

Proof. We know that for any $t, l, t \neq l$: $|E_k - E_l| \leq Q$. Let us suppose that, for some t, l, t, and for some k, we have: $X^t_{p_0(k),k} - X^l_{p_0(k),k}| \geq Q + TP^2 + 1$. Then (4.5) implies the existence of $k' \neq k$ such that $p_0(k') \cdot X^l_{p_0(k'),k'} \geq P^2$, which also implies that $X^l_{p_0(k'),k'} \geq P \geq p_0(k)$. By the same way, we clearly have $X^t_{p_0(k),k} \geq P \geq p_0(k')$. So we modify current vector X as follows:

- Make $X^t_{p_0(k),k}$ decrease by $p_0(k')$ and $X^l_{p_0(k),k}$ increase by $p_0(k')$;
- Make $X^t_{p_0(k'),k'}$ increase by $p_0(k)$ and $X^l_{p_0(k'),k'}$ decrease by $p_0(k)$.

Doing this does not modify values $\sum_{t,p} c_{pk} X^t_{pk}$ and H_t and so maintains constraints (4.2), (4.5) as well as the cost value. It also maintains constraints (4.4). Let us now check that we can iteratively perform this procedure until getting (4.6). In order to do it, we choose t, l such that $X^t_{p_0(k),k} - X^l_{p_0(k),k}$ is maximal. Then we choose k' such that $X^l_{p_0(k'),k'} \geq X^t_{p_0(k'),k'}$. We can do it because of (4.5). Once above procedure has been applied to X, at least one of the two following conditions holds:

- The value $\sup_{k,t,l \text{ s.t. } t \neq l} X^t_{p_0(k),k} - X^l_{p_0(k),k}$ have decreased;
- Above value remains the same, but the number of 3-uples (t, l, k) which achieve $\sup_{k,t,l \text{ s.t. } t \neq l} X^t_{p_0(k),k} - X^l_{p_0(k),k}$ have decreased.

This allows to conclude. □

As previously told, the upper bounds Q and $Q + TP^2$ involved in lemmas 4.2.2 and 4.2.3 are polynomial functions of the encoding size of our **Multi_Bot** instances. This will provide us in Section 4.3 with the key for the design of the time-polynomial dynamic programming algorithms for the exact resolution of **Multi_Bot** when $K = 1$ and $T = 1$.

Those algorithms will work while controlling the way the sum $\sum_{k,t} X^t_{p_0(k),k}$ evolves and doing in such a way that even if this quantity may become very large, its difference with some pre-computed lower bound remains bounded by a polynomial function of the size of the instances.

4.2.3 The Multi_Bot problem is strongly NP-Hard

Section 4.2.1 tells us that we only need to prove that **Multi_Bot_dec** is strongly NP-Complete. We are going to do it in the usual way, by checking that **Multi_Bot_dec** can be polynomialy reduced to the **Bin_Packing** problem [Garey and Johnson, 1979], or, in other words, that **Bin_Packing** is a particular case of **Multi_Bot_dec.**

This argument involving the **Bin_Packing** problem will not only provide us here with a theoretical result, but also with the basis for the design in Section 4.4 of an efficient heuristic algorithm for the general case.

A **Bin_Packing** instance *BP* is characterized by:

- A set $\mathbf{I} = \{1, \ldots, I\}$ of *items* such that for any item $i \in \mathbf{I}$, is provided with a weight w_i;
- A set $\mathbf{B} = \{1, \ldots, B\}$ of identical *boxe*s, all with a same capacity ρ.

Then we want to compute an assignment σ from **I** to **B**, consistent with the capacities of the boxes, that means such that for any $b \in \mathbf{B}$, $\sum_{i \text{ s.t. } \sigma(i)=b} w_i \leq \rho$.

We know that **Bin_Packing** is strongly NP-Complete: Strongly means that there exists a polynomial function Q of I and B such that if we restrict ourselves to instances *BP* such that weights w_i and the capacity ρ are bounded by $Q(I, B)$, then resulting problem remains NP-Complete [Garey and Johnson, 1979]. In order to prove that **Multi_Bot_dec** is NP-Complete, we only need to design a time-polynomial algorithm *Code* which turns any **Bin_Packing** instance *BP* into a **Multi_Bot_dec** instance *Multi_Bot_dec* = *Code*(*BP*) and a time-polynomial algorithm *Decode* which turns any **Multi_Bot_dec** output X into a **Bin_Packing** output σ in such a way that: *BP* admits a feasible solution σ if and only if *Code*(*BP*) admits a feasible solution X such that *Decode*(X) = σ.

Theorem 4.2.4. ***Multi_Bot_dec*** *is strongly NP-Complete*

Proof. Its ILP formulation implies that **Multi_Bot_dec** is in NP. Thus, we only need to prove that it contains **Bin_Packing** in the sense of the *Code/Decode* reduction scheme.

Let us start from a **Bin_Packing** instance BP such that number ρ and coefficients w_i are all bounded by $Q(I, B)$, with polynomial function Q as above. We notice that if we add $B \cdot \rho - \sum_i w_i$ items with weight 1 then we do not modify the feasibility of BP. So we may suppose that BP is such that:

$$B \cdot \rho = \sum_i w_i. \tag{4.7}$$

Let us now derive from the instance BP a **Multi_Bot_dec** instance *Multi_Bot_dec* as follows:

- We set $t = B$: that means that we identify any box b with an index value t of *Multi_Bot_dec*.
- We set $\alpha = 1$ and $\beta = 1$.
- We set K = Number of distinct values w_i involved into vector w. We identify any existing weigth w_i with an index value k of *Multi_Bot_dec*. K may be interpreted as an item type, characterized by its weight $w(k)$. For instance, if we have 5 items with respective weights $w_1 = 2, w_2 = 6, w_3 = 3, w_4 = 2$ and $w_5 = 6$, then we get $K = \{1, 2, 3\}$, with $w(1) = 2, w(2) = 6, w(3) = 3$.
- We set $P = \sup_i w_i$. We identify any possible weigth w_i with an index value p of *Multi_Bot_dec*.
- According to this, we set: $c_{pk} = w(k)$ if $p = w(k)$ and $c_{pk} = 0$ else.
- For any k, we set $n_k = w(k) \cdot R_k$, where R_k denotes the number of items i with weight $w_i = w(k)$).
- Finally we set $S = (B + 1)\rho$.

According to this, X^t_{pk} refers to the number of items with weigth p and item type k which are assigned to box t. Constraints (4.2) means that all items should be assigned

to some box t. Since (4.7) implies $\sum_{t,p,k} pX^t_{pk} = B\rho$, then (4.3) implies that every sum $H_t = \sum_{p,k} pX^t_{pk}$ should be equal to ρ.

If σ is a feasible solution of the **Bin_Packing** instance BP then we see that we get a feasible solution X of the instance *Multi_Bot_dec* by setting: $X^t_{w(k),k}$ = number of items i with weight $w(k)$ assigned to box t, and $X^t_{pk} = 0$ if $w(k) \neq p$. Conversely, if X is a feasible solution of the instance *Multi_Bot_dec*, then we get a solution σ of the instance BP by assigning $X^t_{w(k),k}$ items with weight $w(k)$ to box t.

The *Code* procedure which computes T, P, K together with coefficients S and $n_k, k \in K$, clearly works in polynomial time (as a function of $Q(I,B)$, I and B and thus also as a function of I and B). The *Decode* procedure which retrieves assignment σ from X works the same way. So we conclude. □

4.3 Exact polynomial algorithms for special cases

Our purpose here is to prove that, in case one of the 3 main parameters K, T, P of the **Multi_Bot** problem is neutralized, that means is equal to 1, then **Multi_Bot** can be solved in polynomial time. Doing it will provide us with tools for the design in Section 4.4 of an efficient heuristic for the **Multi_Bot** problem. Since dynamic programming will be deeply involved, we are first going to recall a few things about dynamic programming.

4.3.1 A few things about dynamic programming

Dynamic programming [Bellman, 1966], abbreviated by DP, can be applied to a problem $\mathscr{P}$ if $\mathscr{P}$ can be rewritten in such a way it becomes equivalent to the search for a shortest or largest path in an acyclic network $\mathbf{G}^{\mathscr{P}}$. Notice that while in most cases, the operator which underlies the notion of length of a path is the standard sum operator $+$, we may in some case deal with $*$, max or min operators. In any case Bellman principle is applied, which allows the computation, for any node x of $\mathbf{G}^{\mathscr{P}}$, of the optimal value of a shortest (largest) path from origin x_0 to x according to the formula: $V(x) = \inf arcs_{e=(y,x)}(V(y) + Cost(e))$. The key issue becomes related to the number of nodes in the network $\mathbf{G}^{\mathscr{P}}$.

In many cases (the simplest ones), the node set of $\mathbf{G}^{\mathscr{P}}$ appears as a set of pairs (j,s), $j = 0, 1, \ldots, N+1$, and s belongs to a set $\mathscr{S}(j)$ of *states* related to *time* value j. $\mathscr{L}$ =

$\{0, \ldots, N+1\}$ is then called the time space and $\mathscr{S}(j)$ is the state space related to j. In such a case, corresponds to any pair (j, s) a *decision* set $\mathscr{D}(j, s)$, as well as a *feasibility* procedure $\mathscr{O}$: $(j, s, d) \rightarrow \mathscr{O}(j, s, d) \in \{0, 1\}$, which checks whether decision $n \in \mathscr{D}(j, s)$ can be applied to s at time j or not. In case n can be applied, it induces a *transition* $\mathscr{T}(j, s, d)$: $(j, s) \rightarrow^d (j+1, s^*)$, provided with some cost $C(j, s, d)$, which underlies an arc $((j, s), (j+1, s^*))^d$ of $\mathbf{G}^{\mathscr{P}}$. Therefore, the goal of a DP algorithm is to compute through Bellman principle [Broumi et al., 2019] a sequence of decisions which will allow moving from some initial state s_0 at time 0 to final state s_f at time $N+1$, with a smallest cumulated cost.

Such an algorithm may be designed according to the following *forward driven* scheme:

Initialization:

- We set $\mathscr{S}(j) = \{(s_0, 0, d_1 = Undefined)\}$, where s_0 is the initial state;
- For any $j \neq 0$, we set $\mathscr{S}(j) = Empty\ list$;

Explanation: For any j, $\mathscr{S}(j)$ provides the set of states s which we reached from s_0, together with their best cumulated cost V and the decision n_1 which allowed us to switch from some former state s' in $\mathscr{S}(j-1)$ to s.

Main Loop:

> Scan time space $\mathscr{L}$, and, for any j, scan the state set $\mathscr{S}(j)$: for any 3-uple (s, V, d_1) in $\mathscr{S}(j)$, generate all feasible decisions n in $\mathscr{D}(j, s)$, together with resulting states s* in $\mathscr{S}(j+1)$ and related cost $C(j, s, d)$. At the time when (s, V, d_1), n and s^* are generated, we are provided with a current set $\mathscr{S}(j+1)$: then we compare s^* and the states s' currently in $\mathscr{S}(j+1)$. If s^* is *dominate*d (Bellman principle) by some 3-uple $(s' = s^*, V', d')$, that means if $V + C(j, s, d) \geq V'$, then we drop s; Else, we insert $(s^*, V + C(j, s, d), d)$ into $\mathscr{S}(j)$ and we remove any 3-uple (s^*, V', d') which could appear in $\mathscr{S}(j+1)$.

The main components of a DP algorithm are:

- The time space $\mathscr{L}$ and, for every j, the state space $\mathscr{S}(j)$;
- For every (j,s), the decision set $\mathscr{D}(j,s)$ and the feasibility procedure $\mathscr{C}$;
- The transition procedure $\mathscr{T}(j,s,d)$, which, to any (j,s) and any feasible decision n in $\mathscr{D}(j,s)$, makes correspond resulting pair $(j+1,s^*)$, together with cost $C(j,s,d)$;
- The search strategy (forward, backward, ...);
- Potential *filtering* devices, which aim at controlling the number of states in $\mathscr{S}(j)$, by killing states s which may be considered as not promising enough.

4.3.2 Pre-processing

This section is dedicated to pre-processing the variables $X^t_{pk}, p \neq p_0(k)$ and building the table *Table_Bot*. As told in Section 4.2, we are going to handle the general **Multi_Bot** problem while focusing on "critical" variables $X^t_{p_0(k),k}$. More precisely, we shall neutralize the other variables $X^t_{pk}, p \neq p_0(k)$, while using a pre-process relying on Lemma 4.2.1 and the fact that every $X^t_{pk}, p \neq p_0(k)$, is bounded by $(p_0(k)/GCD(p_0(k),p)) - 1$. The key for this approach is that for any t,k, if we know in advance the value W of $\sum_{p\neq p_0(k)} pX^t_{pk}$, then the best we can do is to compute values X^t_{pk} in such a way that $\sum_{p\neq p_0(k)} c_{pk}X^t_{pk}$ be the largest possible, that means as an optimal solution of the following Knapsack like problem **Aux_Bot**(k,W):

Aux_Bot(k,W): Compute $Y = (Y_p, p \neq p_0(k))$, integral and such that:

$$\sum_{p\neq p_0(k)} pY_p \leq W; \tag{4.8}$$

For any $p, Y_p \leq (p_0(k)/GCD(p_0(k),p)) - 1$; (*Because of Lemma 4.2.1*)

$V = \sum_{p\neq p_0(t)} c_{pk}Y_p$ is the largest possible.

We are going to check that solving this problem can be done in polynomial time through a single DP algorithm *A_Bot*. *A_Bot* will compute all 3-uples (W, V, Y) such that V and Y define an optimal solution of **Aux_Bot**(k, W) and which are Pareto optimal. Pareto optimal means here that there does not exist $W' < W$ and Y' which is a feasible solution of **Aux_Bot**(k, W') and which achieves a value $V' \geq V$. Then, performing *A_Bot* for any value k and storing resulting 3-uples (W, V, Y) in a list *Table_Bot*[k] will provide us in polynomial time with the vectors $Y = (Y_p, p \neq p_0(k)) = (X^t_{pk}, p \neq p_0(k))$ likely to be involved into an optimal **Multi_Bot** solution. In other terms, every time we must deal some pair (t, k) and decide about values $X^t_{pk}, p \neq p_0(k)$, we shall only decide about value W, next retrieve V and Y from the table *Table_Bot* and finally set $X^t_{pk} = Y_p$ for any $p \neq p_0(k)$.

The algorithm *A_Bot* is a dynamic programming algorithm with the following characteristics:

- The time space $\mathscr{L}$ is the set $\{0\} \cup \{p = 1, \ldots, P, p \neq p_0(k))$; The successor $Succ(p)$ of a time value p is $p + 1$ in case $p + 1 \neq p_0(k)$ and $p + 2$ else.
- The state space $\mathscr{S}$ is the set of all non negative number w such that $w \leq (p_0(k) - 1) \sum_{p \neq p_0(k)} p$. Every state w will be given together its cumulated *profit* (we seek maximization) v.
- A decision n at time p is a non negative integral number between 0 and $p_0(k)/GCD(p, p_0(k)) - 1$, which means at time p the value of $Y_{Succ(p)}$.
- Related transition increases w by $Succ(p) \cdot d$ and cumulated *profit* v by $c_{Succ(p),k} \cdot d$.
- Initial state is 0 with related value 0; Related profit v is 0.
- Final states are all W which could be reached at time P, provided with a final cumulative profit V.
- Bellman principle: for any p, we only keep Pareto pairs (w, v), which means that we forbid two pairs (w, v) and (w', v') related to a same set $\mathscr{S}(p)$ to be such that $w \leq w'$ and $v \geq v'$.

The whole algorithmic scheme is a mere adaptation of the general DP scheme of Section

4.3.1:

Algorithm 1 *A_Bot*(k)

1: *Initialize:*

2: $\mathscr{S}(0)$ = a list reduced to one 3-uple $(w = 0, v = 0, d = Undefined)$

3: For any $p \geq 1$ and $\neq p_0(k)$, $\mathscr{S}(p) = Empty\ list$;

4: *Main loop:*

5: Scan the time space $\mathscr{L}$ and, for any $p \neq P$ in $\mathscr{L}$ (or $\neq P-1$ in case $P = p_0(k)$) scan current set $\mathscr{S}(p)$:

For any $(p, (w, v, d_1))$ generated this way, scan the decision set $n \leq p_0(k)/GCD(p, p_0(k)) - 1$:

For any n generated this way, compare the 3-uple $(v + d \cdot Succ(p), w + c_{Succ(p),k} \cdot d, d)$ with the elements of current list $\mathscr{S}(Succ(p))$:

If there exists (w', v', d') in $\mathscr{S}(Succ(p))$ such that $w' \leq w$ and $v' \geq v$ then drop (w, v, d);

Else insert (w, v, d) into $\mathscr{S}(Succ(p))$ while removing from $\mathscr{S}(Succ(p))$ all 3-uples (w', v', d') such that $w \leq w'$ and $v \geq v'$;

6: *Output:*

7: For any (w, v, d) in $\mathscr{S}(P)$ (or $\mathscr{S}(P-1)$) if $P = p_0(k)$, retrieve solution vector Y, and put related 3-uple (w, v, Y) into the list *Table_Bot*[k].

Lemma 4.3.1. *A_Bot(k) solves Aux_Bot*(k, W) *in polynomial time*

Proof. *A_Bot*(k) clearly solves *Aux_Bot*(k, W). The fact that it does it in polynomial time is a mere consequence of Lemma 4.2.1 and the constraints: For any $p, Y_p \leq (p_0(k)/GCD(p_0(k), p)) - 1$. Then the number of states w is bounded by $(p_0(k) - 1) \sum_{p \neq p_0(k)} p$, which is a polynomial function of P. □

4.3.3 Single type of load

Since $K = 1$, we may simplify our notations as follows: X^t_{pk} becomes X^t_p, c_{pk} becomes c_p, n_k becomes n, and $p_0(k)$ becomes p_0. Then the problem becomes:

Multi_Bot_K_1: Compute $X = (X_p^t, p = 1, ..., P, t = 1, ..., T)$ such that:

$$\sum_{t,p} c_p X_p^t \geq n; \tag{4.9}$$
$$\alpha(\max_t \sum_p p X_p^t) + \beta(\sum_{t,p} p X_p^t) \text{ is the smallest possible.}$$

The idea for the algorithm is the following. As suggested in the previous sections, we shall rely here on a dynamic programming algorithm, with a time space which follows the values $t = 1, \ldots, T$. The question is clearly about the definition of the state space, since values $X_{p_0}^t$ and c_{p_0} may not be bounded by any polynomial function of our **Multi_Bot** inputs. In order to overcome this difficulty, we shall define a state at time t while splitting it into 2 components:

- A component made of a number W whose meaning will be the value $\sum_{l \leq t, p \neq p_0} p X_p^l$ related at time t to the contribution of the variables $X_p^l, p \neq p_0$. We are going to use Lemma 4.2.1 and the table *Table_Bot* pre-computed in Section 4.3.2 in order to manage those variables $X_p^l, p \neq p_0$ as a whole according to a single decision.

- A component EX which will refer to the difference between $\sum_{l \leq t} p_0 \cdot X_{p_0}^l$ and some number $UMin(t)$. The key here is that we are going to use lemmas 4.2.2 and 4.2.3 in order to identify (next lemmas 4.3.3, 4.3.4 and 4.3.5) 2 numbers $UMin(t)$ and $UMax(t)$ whose difference is a polynomial function of our **Multi_Bot** inputs and which are such that $\sum_{l \leq t} p_0 \cdot X_{p_0}^l$ may be maintained between $UMin(t)$ and $UMax(t)$.

Related decisions will be designed accordingly.

Let us now enter into the details of this approach, and state:

Theorem 4.3.2. ***Multi_Bot_K_1*** *may be solved in polynomial time*

Proof. It is enough (Section 4.2.1) to prove that the decisional version of **Multi_Bot_K_1** can be solved in polynomial time. S being such that $S \leq 2 \sup(\alpha, \beta) TPn$, (see Section 4.2.1), **Multi_Bot_dec_K_1**(S) comes as:

Multi_Bot_dec_K_1(S): Compute $X = (X_p^t, p = 1, ..., P, t = 1, ..., T)$ such that:

$$\sum_{t,p} c_p . X_p^t \geq n; \tag{4.10}$$

$$\alpha(\max_t \sum_p p X_p^t) + \beta(\sum_{t,p} p X_p^t) \leq S. \tag{4.11}$$

We are going to design a polynomial time DP algorithm with time space $0, 1, \ldots, T$, which solves **Multi_Bot_dec_K_1.** But in order to control the number of states of this algorithm, we must restrict the search space of **Multi_Bot_dec_K_1** , which we are going to do through the following lemmas:

Lemma 4.3.3. *Given a **Multi_Bot_dec_K_1** instance mb_dec_K_1, reinforced by* (4.4), (4.5), (4.6). *Then imposing the following constraint* (4.12) *does not modify the feasibility value of mb_dec_K_1:*

$$\alpha \cdot \sup_t H_t + \beta \cdot H \geq S - (\alpha + T\beta). \tag{4.12}$$

Proof. Given a feasible solution X of *mb_dec_K_1*. We see that if we increase every value $X_{p_0}^t$ by 1, then we do not lose any of the constraints (4.9), (4.4), (4.5), (4.6). So we may impose the following constraint (4.13):

$$X \text{ is maximal in the sense that increasing every } X_{p_0}^t \text{ by 1 violates } (4.11). \tag{4.13}$$

But if X satisfies (4.13) then we must have: $\alpha \cdot \sup_t H_t + \beta \cdot H \geq S - (\alpha + T\beta)$. So we conclude. □

Lemma 4.3.4. *For any $t_0 = 1, \ldots, T$, we have $\sum_{t \leq t_0} p_0 X_{p_0}^t \leq t_0 S/(\alpha + T\beta) + t_0 Q$, with Q defined in Section 4.2.2*

Proof. We know (Lemma 4.2.2) that for any pair $t, l, t \neq l$, $|p_0 X_{p_0}^t - p_0 X_{p_0}^l| \leq Q$. Let us set $\phi = \inf_t p_0 X_{p_0}^t$. (4.11) implies: $(\alpha + T\beta)\phi \leq S$ and so, for any t, $p_0 X_{p_0}^t \leq \phi + Q \leq S/(\alpha + T\beta) + Q$. It comes that for any t_0, $\sum_{t \leq t_0} p_0 X_{p_0}^t \leq t_0 Q + t_0 S/(\alpha + T\beta)$. □

Lemma 4.3.5. *For any $t_0 = 1, \ldots, T$, we must have $\sum_{t \leq t_0} p_0 X_{p_0}^t \geq t_0 S/(\alpha + T\beta) - t_0(2Q + 1 - p_0)$.*

Proof. Let us denote by t^* the index value t which achieves $\sup_t H_t$. (4.12) tells us that $\alpha H_{t^*} + \beta \sum_t H_t \geq S - (\alpha + K\beta)$. But we may decompose $\alpha H_{t^*} + \beta \sum_t H_t$ as $\alpha p_0 X_{p_0}^{t^*} + \beta \sum_t p_0 X_{p_0}^t + \alpha F_{t^*} + \beta \sum_t F_t$. We know that $\alpha F_{t^*} + \beta \sum_t F_t \leq (\alpha + K\beta)(Q - p_0)$. We deduce $\alpha p_0 X_{p_0}^{t^*} + \beta \sum_t p_0 . X_{p_0}^t \geq S - (\alpha + T\beta) - (\alpha + T\beta)(Q - p_0) = S - (\alpha + T\beta)(Q - p_0 + 1)$. Let us set $\phi = \inf_t p_0 X_{p_0}^t$ as in Lemma 4.3.4. We deduce $(\alpha + T\beta)(Q + \phi) \geq \alpha p_0 X_{p_0}^{t^*} + \beta \sum_t p_0 X_{p_0}^t \geq S - (\alpha + T\beta)(Q - p_0 + 1)$ and so $(\alpha + T\beta)\phi \geq S - (\alpha + T\beta)(2Q - p_0 + 1)$. So we get that, for any t, $p_0 X_{p_0}^t \geq S/(\alpha + T\beta) - (2Q - p_0 + 1)$ and the result. □

For any t, let us set $UMin(t) = t_0 S/(\alpha + T\beta) - t_0(2Q + 1 - p_0)$ and $UMax(t) = t_0 S/(\alpha + T\beta) + t_0 Q$. The key point is that the number of possible integral values $\sum_{t \leq t_0} p_0 X_{p_0}^t$ becomes bounded by $UMax(t) - UMin(t)$, which itself bounded by a polynomial function of T and P. By the same way we know that $\sum_{l \leq t} \sum_{p \neq p_0} p X_p^l$ is also going to be bounded by a polynomial function of T and P (Lemma 4.2.2). So we become able to provide the main components of the DP algorithm *Multi_Bot_dec_K_1(S)*:

- **Time Space**: The set of all values $t = 0, 1, \ldots, T$.
- **State Space**: A state at time t is given by:
  - A number W whose meaning is $\sum_{l \leq t} \sum_{p \neq p_0} p X_p^l$ induced by the decisions taken until time t;
  - The value EX whose meaning is the difference $(\sum_{l \leq k} p_0 . X_{p_0}^l - UMin(t))$, whose value is non negative and no larger than $UMax(t)$ because of lemmas 4.3.1, 4.3.3, 4.3.4.
  - Related Cost value: To any state (W, EX) at time t corresponds some current cost VAL which means current value $\sum_{l \leq t} \sum_p p X_p^l$;
  - Initial state at time 0 is (0, 0). Related value VAL is 0.
- **Decisions at time** $t = 0, \ldots, T - 1$:
  - We choose a 3-uple (w, v, Y) in the list *Table_Bot*: for any $p \neq p_0$ we set $X_p^{t+1} = Y_p$;
  - We decide some number z between 0 and $(UMax(t+1) - UMin(t+1))$; Related value $X_{p_0}^{t+1}$ is going to comes as $(z + UMin(t+1) - (EX + UMin(t))/p_0$.

– Feasibility criterion: we should have:

* $\alpha(z + UMin(t+1) - (EX + UMin(t)) + \beta(W + w) \leq S$;

* $(z + UMin(t+1) - (EX + UMin(t))$ is an integral multiple of p_0;

* $0 \leq (z + UMin(t+1) - (EX + UMin(t))$.

- **Transitions**:

  W is turned into $W + w$;

  EX becomes z;

  VAL is incremented by $c_{p_0}(z + UMin(t+1) - (EX + UMin(t)) + v$.

- **Bellman Principle**: There must not exists 2 states (W, EX) and (W', EX') in $\mathscr{L}(t+1)$ such that $W \leq W'$, $EX \leq EX'$ and $VAL \geq VAL'$, VAL and VAL' being the cost value related to (W, EX) and (W', EX') respectively.

The fact that this dynamic programming algorithm solves **Multi_Bot_dec_K_1**(S) derives from Lemma 4.2.2 and lemmas 4.3.3, 4.3.4, 4.3.5 and from the fact that any pair (W, EX) determines the whole values $\sum_{l \leq t} \sum_{p \neq p_0} pX_p^l$ and $\sum_{l \leq t} p_0 X_{p_0}^l$. In order to achieve our proof, we only need to check that it works in polynomial time. We first notice that the number of possible state values W is polynomial bounded in T and P. The same holds for values EX, because the difference $UMax(t) - UMin(t) = t_0 Q + t_0(2Q + 1 - p_0)$ depends on T and P in a polynomial way. The number of possible decisions w contained into the list *Table_Bot* is also bounded by a polynomial function of T and P, and the same holds for the number of decisions z. We conclude. □

4.3.4 Single period

Since $T = 1$, we simplify our notations as follows: X_{pk}^t becomes X_{pk}. By the same way, we set: $H = \sum_{kp} pX_{pk}; E = \sum_k p_0(k) X_{p_0(k),k}; F = \sum_{k, p \neq p_0(k)} pX_{pk}$.

Then **Multi_Bot** becomes:

Multi_Bot_T_1: Compute $X = (X_{pk}, p = 1, ..., P, k = 1, ..., K)$ such that:

$$\text{For any } k, \sum_{p} c_{pk} X_{pk} \geq n_k;$$

$$\sum_{k,p} p X_{pk} \text{ is the smallest possible.}$$

The idea for the algorithm: as suggested in the previous sections, we shall rely here on dynamic programming, with a time space which follows the values $k = 0, \ldots, K$. Once again, the question is about the definition of the state space, since values $X_{p_0,k}$ and $c_{p_0,k}$ may take very large values. In order to overcome this difficulty, we are going to define a state at time k while splitting it into 2 components:

- A component made of a number W whose meaning will be the value $\sum_{u \leq k, p \neq p_0(u)} p \cdot X_{p,u}$ related at time k to the contribution of the variables $X_{p,u}, p \neq p_0(u), u \leq k$. We are going to use Lemma 4.2.1 and the table *Table_Bot* pre-computed in Section 4.3.2 in order to manage those variables $X_{p,u}, p \neq p_0(u)$ as a whole according to a single decision.

- A component EX which will refer to the difference between $\sum_{u \leq k} p_0 \cdot X_{p_0,u}$ and some number $Min1(t)$. The key here is that we are going to use lemmas 4.2.2 and 4.2.3 in order to identify (next Lemma 4.3.7) 2 numbers $Min1(k)$ and $Max1(k)$ whose difference is a polynomial function of our **Multi_Bot** inputs and which are such that $\sum_{u \leq k} p_0 \cdot X_{p_0,u}$ may be maintained between $Min1(k)$ and $Max1(k)$.

Related decisions will be designed accordingly.

Let us now enter into the details of this approach, and state:

Theorem 4.3.6. ***Multi_Bot_T_1*** *may be solved in polynomial time*

Proof. It is enough to prove that the decisional version of **Multi_Bot_T_1** can be solved in polynomial time. S being a parameter such that $S \leq 2 \sup(\alpha, \beta) PK(\sup_k n_k)$, **Multi_Bot_dec_T_1** may be written:

Multi_Bot_dec_T_1(S): Compute $X = (X_{pk}, p = 1, ..., P, k = 1, ..., K)$ such that:

$$\sum_{p} c_{pk} X_{pk} \geq n_k \quad \forall k \tag{4.14}$$

$$\sum_{k,p} p X_{pk} \leq S \tag{4.15}$$

We are going to design a polynomial time DP algorithm with time space $0, 1, \ldots, K$ and decisions related, for any k, to values $X_{pk}, p = 1, \ldots, P$. As a matter of fact, since values $X_{pk}, p \neq p_0(k)$ may be controlled through the table *Table_Bot*, we focus on the key value $Z_k = X_{p_0(k),k}$. For any k, let us set:

- $Max1(k) = \sum_{u \leq k} p_0(u) \cdot \left\lceil n_u / c_{p_0(u),u} \right\rceil$;
- $Min1(k) = \sum_{u \leq k} p_0(u) \cdot \left\lfloor n_u / c_{p_0(u),u} \right\rfloor - \sum_{u \leq k} (Q(u) - p_0(u))$.

Then we state the following lemma, which is going to help us in controlling the way values $Z_k = X_{p_0(k),k}$ are going to evolve with time value k.

Lemma 4.3.7. *Given a solution X of **Multi_Bot_dec_T_1** which satisfies (4.4), and which minimizes $\sum_{k,p} p X_{pk}$. Let us set, for any k, $U_k = \sum_{u \leq k} p_0(u) X_{p_0(u),u} = \sum_{u \leq k} p_0(u) Z_u$. Then we have, for any k: $Min1(k) \leq U_k \leq Max1(k)$*

Proof. Clearly, $Z_k = X_{p_0(k),k}$ should not exceed $\left\lceil n_k / c_{p_0(k),k} \right\rceil$. Thus, $\sum_{u \leq k} p_0(u) Z_u$ should not exceed $Max1(k)$. On another side, we should have that $c_{p_0(k),k} Z_k + (p_0(k) - 1)(\sum_{p \neq p_0(k)} c_{pk}) \geq n_k$ for any k, because of (4.4). That means that Z_k should be no smaller than $\left\lfloor n_k / c_{p_0(k),k} \right\rfloor - (p_0(k) - 1)(\sum_{p \neq p_0(k)} c_{pk} / c_{p_0(k),k})$, and so that $p_0(k) Z_k$ should not be smaller than $p_0(k) \left\lfloor n_k / c_{p_0(k),k} \right\rfloor - (p_0(k) - 1) p_0(k) (\sum_{p \neq p_0(k)} c_{pk} / c_{p_0(k),k})$. But the definition of $p_0(k)$ implies $c_{pk} / c_{p_0(k),k} \geq p / p_0(k)$ for any p. We deduce that $p_0(k) Z_k$ should be at least equal to $p_0(k) \left\lfloor n_k / c_{p_0(k),k} \right\rfloor - (p_0(k) - 1)(\sum_{p \neq p_0(k)} p) = \left\lfloor n_k / c_{p_0(k),k} \right\rfloor - (Q(k) - p_0(k))$. We conclude. □

We may now design our dynamic programming algorithmic scheme, while specifying its main components:

- **Time Space**: $k = 0, \ldots, K$.
- **State Space**: A state at time k is a pair (W, EX) with the following meaning:
  - $W = \sum_{p \neq p_0(u), u \leq k} p X_{pu}$;

– $EX = \sum_{u \leq k} p_0(u) Z_u - Min1(k)$.

Explanation: W means the amount of values $pX_{pu} \leq (p_0(u) - 1)$ which have been decided until k and which are not related to $p_0(u)$ values. EX provides us with the location, inside the window $\{Min1(k), \ldots, Max1(k)\}$ of the sum of values $p_0(u)X_{p_0(u),u}$ decided until k. Defining the states this way will ensure that the number of states remains bounded by a polynomial function of P and K.

- **Initial state**: (0, 0): no decision has been taken.
- **Final State**: Any pair (W, EX) related to time value K.
- **Decisions**: At any $k = 0, \ldots, K-1$, and for any current state (W, EX) we choose:
  - A 3-uple $(w, v = \sum_{p \neq p_0(k+1)} c_{p,k+1} X_{p,k+1}, Y)$ in *Table_Bot*[k+1], as computed in Section 4.3.2.
  - A value z which between $Min1(k+1)$ and $Max1(k+1)$.

    Explanation: Decision w means that we follow Section 4.3.2 and choose values $X_{p,k+1}, p \neq p_0(k+1)$ while using the table *Table_Bot*. *Table_Bot*[k+1] provides us with the 3-uple (w, v, Y) such that v is the optimal value of the **Aux_Bot**$(k+1, w)$ instance of **Aux_Bot** related to w and $k+1$. As for z, it corresponds to the increment of the new difference $\sum_{u \leq k+1} p_0(u) Z_u - Min1(k+1)$ when we shift to $k+1$, and refers to a quantity $p_0(k+1)X_{p_0(k+1),k+1} = z - Min1(k+1) + Min1(k) - EX$.
- **Feasibility of a decision**:
  - $z - Min1(k+1) + Min1(k) - EX$ must be non negative and a multiple of $p_0(k)$;
  - $v + c_{p_0(k+1),k+1}(z - Min1(k+1) + Min1(k) - EX)/p_0(k)$ should be at least equal to n_k;
  - $W + z + Min1(k+1)$ should not exceed S.
- **Transitions**: Transition from k to $k+1$ works as follows:
  - W becomes $W + w$;

– EX becomes z.

- **Bellman Principle**: We keep, for any time value k, states (W, EX) and related values $\sum_{u \leq k,p} c_{pu} X_{pu}$ which are not dominated in the sense of Pareto.

Checking that above dynamic programming solves **Multi_Bot_dec_T_1** is nothing but routine. As for its complexity, we already saw in Section 4.3.2 that values W and w were bounded by Q, which is a polynomial function of P and K. As for state values EX, we see that the difference $Max1(k) - Min1(k)$ is bounded by a a polynomial function $R(P, K)$. It comes that both the number of state values EX and decision values z are also bounded by $R(P, T)$, and so are the number of states (W, EX) and decisions (w, z). We conclude. □

4.3.5 Single robot configuration

This last case does not involve dynamic programming. Since $P = 1$, we simplify our notations as follows: c_{pk} becomes c_k and X^t_{pk} becomes X^t_k. Then our **Multi_Bot** problem becomes:

Multi_Bot_P_1: Compute $X = (X^t_k, t = 1, \ldots, T, k = 1, \ldots, K)$ such that:

$$\text{For any } k, c_k \cdot (\sum_t X^t_k) \geq n_k;$$

$$\alpha(\max_t \sum_k X^t_k) + \beta(\sum_{t,k} X^t_k) \text{ is the smallest possible.}$$

Then we state:

Theorem 4.3.8. ***Multi_Bot_P_1*** *may be solved in polynomial time*

Proof. Section 4.2.1 tells us that it is enough to prove that the decisional version of **Multi_Bot_P_1** may be solved in polynomial time. S being a parameter such that $S \leq 2 \sup(\alpha, \beta) KT(\sup_k n_k)$, (see Section 4.2.1), this decisional version may be written:

Multi_Bot_dec_P_1: Compute $X = (X_k^t, t = 1, \ldots, T, k = 1, \ldots, K)$ such that:

$$\alpha(\max_t \sum_k X_k^t) + \beta(\sum_{t,k} X_k^t) \leq S;$$

$$\text{For any } k, c_k \cdot (\sum_t X_k^t) \geq n_k.$$

For any k, let us set $n_k^* = \lceil n_k / c_k \rceil$. If we impose X to minimize $\sum_{t,k} X_k^t$, then we get that for any k, $(\sum_t X_k^t) = n_k^*$. Let us set $S^* = \lfloor (S - \beta \sum_k n_k^*)/\alpha \rfloor$. Then **Multi_Bot_dec_P_1** becomes:

Multi_Bot_dec_P_1: Compute $X = (X_k^t, t = 1, \ldots, T, k = 1, \ldots, K)$ such that:

$$\text{For any } t, \sum_k X_k^t \leq S^*;$$

$$\text{For any } k, \sum_t X_k^t = n_k^*.$$

At this point, **Multi_Bot_dec_P_1** is no more than a *Transportation* problem, i.e. an ILP with a totally unimodular underlying constraint matrix related to a complete bipartite graph. We conclude. □

4.4 A heuristic algorithm for the general case

We propose now an algorithm *H_Bot* for the general case, which relies on the arguments used in order to get above theoretical results.

This algorithm starts by capitalizing on the knowledge provided by Lemma 4.2.1 and the proof of Theorem 4.2.4. It restricts itself to the handling of critical variables $X_{p_0(k),k}^t$ while interpreting the **Multi_Bot** problem as a variant of the **Bin Packing** problem. According to this purpose, it fills the **Bin Packing** boxes while considering the items according to decreasing weight and assigning them boxes in such a way that the maximal load of a box increases the slowest possible. Next, it takes advantage of Section 4.3 involving dynamic programming techniques, which led in Section 4.3.2 to the pre-computation of the table *Table_Bot*. That means that, for any k, t it redistributes part of the current values $X_{p_0(k),k}^t$ among the variables $X_{p,k}^t$, p different from $p_0(k)$. It does it while using the table

Table_Bot in order to manage those variables $X_{p,k}, p \neq p_0(k)$ as a whole according to a single decision.

More precisely, it works in two steps as described below:

- First step: We solve the specific **Multi_Bot** instance involved in the proof of Theorem 4.2.4 in order to get an initial solution. This specific instance, which only involves variables $X^t_{p_0(k),k}, t \in \mathbf{T}, k \in \mathbf{K}$, is a kind of **Bin_Packing** instance, with item set **K** and box set **T**, which we handle by considering the items according to decreasing weights and assigning them to the box with the smallest current load.

- Second step: It involves the table *Table_Bot* of Section 4.3.2, adapted I such a way that we may control the number of states generated by $A_Bot(k)$and maintain it below some threshold parameter *State_Max*. It aims at improving the solution obtained through step 1 while iteratively applying a local procedure which acts by picking up some pair (t_0, k_0), some 3-uple (w_0, v_0, Y^0) and redistributing part of the value $X^{t_0}_{p_0(k_0),k_0}$ among variables $X^{t_0}_{p,k_0}, p \neq p_0(k_0)$ in order to make the value of the objective function decrease.

The procedure *A_Bot* is rewritten into a procedure $A\_Bot(k, State\_Max)$, where $State\_Max \leq 3P$ is a threshold parameter, in such a way that this procedure computes output list *Table_Bot*[k] exactly as in Section 4.3.2, with the restriction that a state value w is forbidden from exceeding *State_Max*. This impose an additional test on the feasibility of decision n, which must be such that $n + w$ must not exceed *State_Max*. This allows us to better control the running time of the procedure *A_Bot*, which becomes a heuristic for the problem **Aux_Bot**.

4.4.1 First step

As previously told, the first step of *H_Bot* works by focusing on the variables $X^t_{p_0(k),k}$ and making them increase until constraint (4.2) is satisfied in such a way that the values $H_t = \sum_k p_0(k) X^t_{p_0(k),k}, t = 1, \ldots, T$, remain balanced.

Algorithm 2 *H_Bot_First_Step*

1: Order values k according to decreasing $p_0(k)$ values and denote by L^K resulting list;

2: For any t initialize H_t as 0;

3: For any t, k initialize $X^t_{p_0(k),k}$ as 0;

4: For k in L^K do

$B \leftarrow \left\lceil n_k / c_{p_0(k),k} \right\rceil$; /*$B$ measures the gap between current situation and (4.2) satisfaction*/

While $B \geq 1$ do

Pick up t_0 such that H_{t_0} is minimal;

$H_{t_0} \leftarrow H_{t_0} + p_0(k)$;

$X^{t_0}_{p_0(k),k} \leftarrow X^{t_0}_{p_0(k),k} + 1$;

$B \leftarrow B - 1$;

4.4.2 Second step

The second Step of *H_Bot* relies on the following compact representation of a **Multi_Bot** solution. We consider here a **Multi_Bot** solution as given by:

- A vector $Z = (Z^t_k, t = 1, \ldots, T, k = 1, \ldots, K)$, which provides the $X^t_{p_0(k),k}$ values;
- A vector $\Delta = (\Delta^t_k, t = 1, \ldots T, k = 1, \ldots, K)$, where every Δ^t_k is a 3-uple (w^t_k, v^t_k, Y^t_k), belonging to *Table_Bot*[k].

Such a pair (Z, Δ), with indexation on $T \cdot K$, gives rise to a **Multi_Bot** solution X through the formulas:

- For any t, k, p: If $p \neq p_0(k)$ then $X^t_{pk} = (Y^t_k)_p$ else $X^t_{pk} = Z^t_k$.

So *H_Bot_Second_Step* starts from the values $Z^t_k = X^t_{p_0(k),k}$ computed by *H_Bot_First_Step*. Then, for every pair (t_0, k_0), it picks up (w_0, v_0, Y^0) in *Table_Bot*[k_0] in such a way that decreasing $Z^{t_0}_{k_0}$ by $\left\lfloor (c_{p_0(k_0),k_0}(\sum_t Z^{t_0}_{k_0}) - n_{k_0} + v_0) / c_{p_0(k_0),k_0} \right\rfloor$ and setting $X^{t_0}_{p,k_0} = (Y^{t_0}_{k_0})_p = (Y^0)_p$ for any $p \neq p_0(k_0)$ keeps the constraints and yields the best decrease of $\alpha \cdot H^{Max} + \beta \cdot H$.

Algorithm 3 *H_Bot_Second_Step*

1: Start from values $Z_k^t = X^t_{p_0(k),k}$ computed according to *H_Bot_First_Step* and from a null vector Δ;

2: Initialize a vector $MARK = (MARK_k^t, t = 1, \ldots, T, k = 1, \ldots, K)$ with Boolean values, to the null vector;

3: Initialize a vector $NMARK = (NMARK^t, t = 1, \ldots, T)$ with integral values all equal to K;

4: For any k, set $B_k = c_{p_0(k),k}.(\sum_t Z_k^t)$;

5: For any t, set $H_t = \sum_k p_0(k) Z_k^t$; Set $H = \sum_t H_t$;

6: Initialize counter value $COUNT$ to $T \cdot K$;

7: While $COUNT \neq 0$ do

Choose t_0 and k_0; (I1)

For any (w, v, Y) in *Table_Bot*[k_0] compute:

$\gamma = \left\lfloor (B_{k_0} - n_{k_0} + v)/c_{p_0(k_0),k_0} \right\rfloor$;

$HAux = H + w - \gamma \cdot p_0(k_0)$;

$HKAux = H_{t_0} + w - \gamma \cdot p_0(k_0)$;

$HMax = Max(Max_{t \neq t_0} H_t, HKAux)$;

$H^* = \beta \cdot HAux + \alpha \cdot HMax$;

Choose (w_0, v_0, Y^0) such that $\gamma \leq Z_{k_0}^{t_0}$ and which provides us with the smallest value H^*; (I2)

Update:

$MARK_{k_0}^{t_0}$ is set to 1; $COUNT$ and $NMARK^{t_0}$ are decremented by 1;

B_{k_0} is set to $B_{k_0} - \gamma c_{p_0(k_0),k_0} + v_0$;

H_{t_0} is set to $HKAux$ and H is set to $HAux$;

$Z_{k_0}^{t_0}$ is decremented by γ and $\Delta_{k_0}^{t_0}$ becomes (w_0, v_0, Y^0).

We now provide some details for instructions (I1) and (I2).

Instruction (I1)

- Choice of t_0: We target t_0 such that $NMARK^{t_0} \neq 0$ and $H_{t_0} = Max_t H_t$; In case

such a value t does not exist, then we pick up t_0 randomly;

- Choice of k_0: Once t_0 has been chosen, we target k_0 such that $MARK_{k_0}^{t_0} = 0$ and that $p_0(k_0)$ is the largest possible.

Instruction (I2)

- The 3-uple (w^*, v^*, Y^*) may be null, which yields no improvement of our current solution (Z, Δ).

We may implement a variant of the algorithm *H_Bot_Second_Step* by making the main loop depend on a STOP signal, activated when no (t, k) pair exists which induces an improvement of the solution according to above instruction (I2).

4.5 Numerical experiments

Recall that the fleet size (= number of robots) is given by $H^{Max} = \max_t \sum_{k,p} p \cdot X_{pk}^t$ and the number of trips by $H = \sum_{t,p,k} p \cdot X_{pk}^t$. The objective is to minimize $Cost = \alpha \cdot H^{Max} + \beta \cdot H$.

Purpose: We follow here a two-sided purpose. On one side, we want to observe some characteristics of the solutions of our **Multi_Bot** problem, namely, the size H^{Max} of the robot fleet with regard to $Cost$. On the other side, we are interested by the performances of the algorithm *H_Bot*, and more specifically to its ability to compute good solutions (gap to optimality).

Technical Features: Algorithms were implemented in C++, on PC AMD Opteron 2.1GHz, while using gcc 4.1 compiler. On the other hand, to solve the ILP a program was written in Python and solved using PuLP library. PuLP is an application programming interface and it can generate data file created in the Mathematical Programming System (MPS) format or Linear Programming (LP) file and call one of the solvers (GLPK, CBC, CPLEX) to solve linear problems. In the program we use so the default solver is CBC (Coin-or

branch and cut).

Instances: We have generated the instances as follows:

- Decide about T, P, K;
- Randomly select α, β in $[\![0; 1000]\!]$. If $\alpha \leq T \cdot \frac{\beta}{2}$, set $\beta = 0$. We do this in order to have H^{Max} not negligible compared to H, otherwise the problem becomes trivial.
- For every k:
  - Select capacities c_{pk} as follows for every p:
    - Randomly select the mean value c_k^{mean} (of coefficients c_{pk}) in $[\![1; 11]\!]$;
    - Randomly select coefficient c_{pk} in $[\![0; 2c_k^{mean} + 1]\!]$;
    - Normalize capacities: $c_{pk} \leftarrow c_{pk} \cdot \frac{c_k^{mean} P}{\sum_p c_{pk}}$;
    - $c_{pk} \leftarrow round(c_{pk})$; *(rounded to the nearest whole number)*
    - $c_k^{mean} \leftarrow round\left(\frac{1}{P} \sum_p c_{pk}\right)$.
  - Randomly select a coefficient $J \in [\![0; 10000]\!]$. Select demand coefficient n_k as $n_k = \lfloor P \cdot J \cdot c_k^{mean} \rfloor$.

For the purpose of this chapter, we present 20 instances, whose characteristics are summarized in Table 6.

Outputs: For every instance, we compute:

- *Characteristics of the solutions*:
  - H^{Max} = Fleet size
  - H = Number of trips
  - $Cost = \alpha \cdot H^{Max} + \beta \cdot H$
- *Behavior of the Algorithm H_Bot*:

Table 6: Instances

Instances	α	β	T	P	K	c_k^{mean} [a]	J
1	642	21	2	2	2	7;11	8
2	791	4	4	2	2	7;10	848
3	465	95	8	2	2	9;2	601
4	344	0	16	2	2	1;1	945
5	847	2	32	2	2	2;3	286
6	161	0	64	2	2	8;9	273
7	833	0	128	2	2	7;4	816
8	60	0	256	2	2	5;6	233
9	428	0	512	2	2	7;8	161
10	33	0	1024	2	2	7;3	23
11	872	0	2048	2	2	1;5	411
12	981	0	4096	2	2	5;9	8544
13	218	16	2	2	4	4;2;7;3	80
14	443	12	2	2	6	2;3;6;7;10;11	89
15	857	96	2	4	4	8;6;7;3	51
16	90	0	4	4	4	3;7;7;8	41
17	116	74	2	4	6	5;3;6;7;1;11	42
18	328	65	4	4	6	9;1;2;7;5;8	9
19	234	72	6	4	6	4;9;2;9;1;10	6
20	545	71	2	6	6	1;7;4;1;6;3	22

[a] c_k^{mean} are listed by semicolon in order: c_1^{mean}; c_2^{mean}; etc.

- GAP_cost = $\frac{Cost(H\_Bot)-Cost(PuLP)}{Cost(PuLP)}$: Gap to optimality with respect to cost
- GAP1_cost = Gap to optimality with respect to cost when we restrict ourselves to the first step of *H_Bot*
- GAP_fleet = $\frac{H^{Max}(H\_Bot)-H^{Max}(PuLP)}{H^{Max}(PuLP)}$: Gap to optimality with respect to fleet size
- GAP1_fleet = Gap to optimality with respect to fleet size when we restrict ourselves to the first step of *H_Bot*

The results provided by our experiments may be summarized into the flowing Tables 7, 8.

Table 7: Results cost

Instances	Cost(*PuLP*)[b]	Cost(*H_Bot*)	Cost(*1_Step*)	GAP_cost	GAP1_cost
1	12975	12975	12975	0%	0%
2	857022	857022	857026	0%	**0.001%**
3	499705	499705	499705	0%	0%
4	81872	81872	81872	0%	0%
5	32780	32780	32780	0%	0%
6	4025	4025	4025	0%	0%
7	29155	29988	29988	**3.0%**	**3.0%**
8	300	300	300	0%	0%
9	856	856	856	0%	0%
10	33	33	33	0%	0%
11	1744	1744	1744	0%	0%
12	-	11772	11772	-%	-%
13	100280	100280	100280	0%	0%
14	324086	324086	324086	0%	0%
15	706450	707307	707445	**0.121%**	**0.141%**
16	31320	31320	31320	0%	0%
17	273768	273884	273884	**0.042%**	**0.042%**
18	54096	54096	55668	0%	**3.0%**
19	-	24894	25200	-%	-%
20	-	671744	671744	-%	-%

[b] the dash means that no result has been obtained after one hour of computation.

For 14 of the 20 instances, the *H_bot* algorithm finds the optimal solution after the first step of the algorithm. For 3 other instances, it does not find the optimal solution and the cost gap does not exceed 3 %. The fleet gap does not exceed 4 % after the first step and 3 % after the second step. For the last 3 instances, the exact method does not find the optimal solution in less than one hour.

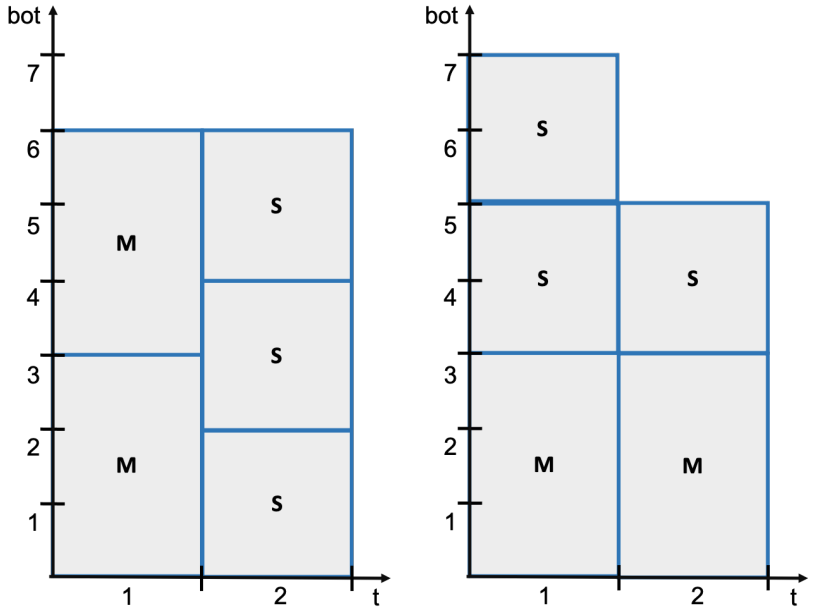
Table 8: Results fleet

Instances	$H^{Max}(PuLP)$[c]	$H^{Max}(H\_Bot)$	$H^{Max}(1\_Step)$	GAP_fleet	GAP1_fleet
1	19	19	19	0%	0%
2	1062	1062	1062	0%	0%
3	408	408	408	0%	0%
4	238	238	238	0%	0%
5	36	36	36	0%	0%
6	25	25	25	0%	0%
7	35	36	36	**3.0%**	**3.0%**
8	5	5	5	0%	0%
9	2	2	2	0%	0%
10	1	1	1	0%	0%
11	2	2	2	0%	0%
12	-	12	12	-%	-%
13	436	436	436	0%	0%
14	694	694	694	0%	0%
15	710	711	711	**0.141%**	**0.141%**
16	348	348	348	0%	0%
17	1037	1038	1038	**0.096%**	**0.096%**
18	92	92	96	0%	**4.0%**
19	-	39	40	-%	-%
20	-	978	978	-%	-%

[c] the dash means that no result has been obtained after one hour of computation.

Note that we can build instances with larger gaps. For instance, take two configurations of robots ($p \in \{2,3\}$), two types of loads ($k \in \{1,2\}$) and two periods ($T = 2$). Loads of type 1, 2 correspond to respectively small (S) and medium (M) loads. The capacities are: There are 3 loads of type 1 ($n_1 = 3$) and 2 loads of type 2 ($n_2 = 2$). The objective is to minimize the number of robots ($\alpha = 1$, $\beta = 0$). A 2-bot can transport only a small load ($c_{21} = 1$, $c_{22} = 0$) and a 3-bot can transport only a medium load ($c_{31} = 1$, $c_{32} = 0$). Then $H^{Max}(PuLP)$ = 6, $H^{Max}(H\_Bot)$ = 7 and GAP_cost = GAP_fleet $\approx$ 16.7%. Figure 37 represents the Gantt chart for this example. On the ordinate axis is the reference number

of the bots, on the abscissa axis is the reference number of the period. The load type the robot is carrying is indicated in each rectangle.



(a) Optimal solution

(b) *H_Bot* solution

Figure 37: Gantt chart for simple example with gap 17%.

Conclusion

We have proved that the problem is strongly NP-complete by reducing it to the Bin Packing problem. We have showed that in three special cases (single period, single load type or single configuration), the problem can be solved in polynomial time with appropriate dynamic programming algorithms. We have then derived from our theoretical results an efficient heuristic algorithm for the general case. A numerical study shows that the heuristic algorithm can successfully be applied even for large instances and has good performances on the tested instances.

Chapter 5

Planning problem

Contents

In this chapter we consider the reconfigurable robots in the context of the problem of scheduling with the objective to minimize the time to execute all transportation tasks. We propose an integer linear programming formulation for the problems with or without reconfiguration. We show that reconfigurability can reduce significantly the execution time.

5.1 Problem description

We consider a fleet of N mobile bots capable of cooperating to transport loads. A p-bot is a set of p bots that are aggregated to carry loads ($p = 1, \cdots, P$). There are n_k loads of type k to be transported ($k = 1, \cdots, K$). A p-bot is able to carry c_{pk} loads of type k. Therefore the poly-robots have a capacity that depends on their configuration and the load type. All loads are to be transported from one point to another in the warehouse. The time horizon is divided into T periods ($t = 1, \cdots, T$). During a period, a p-bot is able to perform a round trip and carry a maximum of c_{pk} loads of type k. The duration of a round trip is independent of the configuration and load type.

The goal is to minimize the makespan, i.e. the time to transport all loads from one place of the warehouse to another. We examine two scenarios. In the first scenario, reconfiguration is allowed, and the configurations can be modified at the beginning of each period. In the second scenario, reconfiguration is not permitted and the configurations are determined for the whole time horizon. The minimum makespan with allowed reconfiguration is denoted T_R. The minimum makespan without any reconfiguration during the process is denoted T_W. To have a feasible solution of this problem, T and N must be sufficiently large.

5.2 Integer linear programming formulation

In this section, we present ILP formulations for the two scenarios. We first assume that reconfiguration is allowed. We use the following decision variables:

- N_{pk}^t : number of p-bots transporting loads of type k in period t;
- α_t : binary variable equal to 1 if any load is transported in period t and to 0 otherwise.

$$T_R = \min \quad \sum_{t=1}^{T} \alpha_t \tag{5.1}$$

subject to :

$$\sum_{t=1}^{T} \sum_{p=1}^{P} c_{pk} \cdot N_{pk}^t \geq n_k \qquad \forall k \tag{5.2}$$

$$\sum_{k=1}^{K} \sum_{p=1}^{P} p \cdot N_{pk}^t \leq N \cdot \alpha_t \qquad \forall t \tag{5.3}$$

$$\alpha_{t+1} \leq \alpha_t \qquad t = 1, \cdots, T-1 \tag{5.4}$$

$$N_{pk}^t \in \mathbb{N},\ \alpha_t \in \{0,1\} \qquad \forall p, \forall k, \forall t \tag{5.5}$$

We now comment each line of the above ILP:

- Objective function (5.1): $\sum_{t=1}^{T} \alpha_t$ represents the makespan;
- Constraint (5.2): all loads of all types must be transported;
- Constraint (5.3): the number of bots used in each period must not exceed the fleet size;
- Constraint (5.4): if period t is inactive, then period $t+1$ is inactive.

When reconfiguration is prohibited, we can simply add the following constraint to the above ILP:

$$\sum_{k=1}^{K} N_{pk}^t \geq \sum_{k=1}^{K} N_{pk}^{t+1} \qquad t = 1, \cdots, T-1,\ p = 1, \cdots, P \tag{5.6}$$

This constraint ensures that there are less p-bots used in period $t+1$ than in period t, which ensures that there is no reconfiguration.

5.3 Reconfigurable vs non-reconfigurable fleet

In this section, we compare the makespans of the two scenarios. Consider the following simple example based on the situation described in Figure 1.

- The first load type corresponds to a box and will be referred to as Small (S);
- The second load type corresponds to a pallet and will be referred to as Large (L);

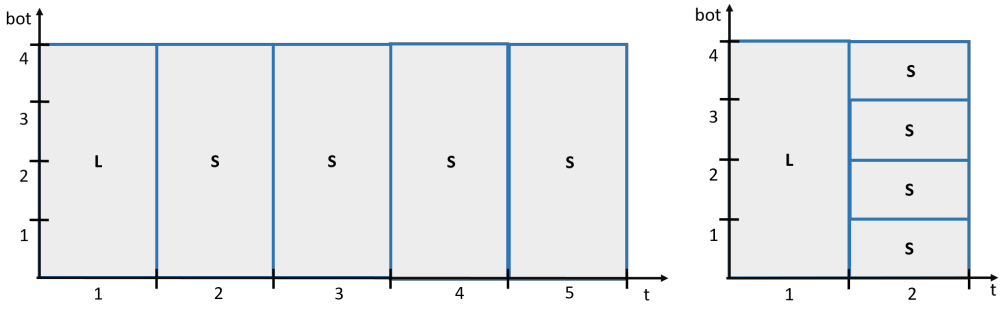
- A 1-bot can carry one box at a time;
- A 4-bot can carry either one pallet at a time or one box at a time (we assume that each box is in a different location and there is no gain to organize a tour);
- There are $N = 4$ bots;
- There are 1 load of type Large and 4 loads of type Small

Table 9 summarizes the capacities of the different configurations.

Configuration	Type of load	
	Small	Large
1-bot	1	0
4-bot	1	1

Table 9: Capacity matrix for 2 types of loads

The optimal solutions for both strategies are shown as Gantt charts in Figure 38. When reconfiguration is allowed, a 4-bot transports a load of size L in period 1 and then splits into four 1-bots in period 2 to transport four loads of size S. The makespan is $T_R = 2$. When reconfiguration is not allowed, the 4-bot transports the load of size L in period 1, then the loads S in periods 2, 3, 4, 5. The makespan is $T_W = 5$. We conclude that the strategy with reconfiguration is 2.5 times faster than the strategy without reconfiguration.



(a) Without reconfiguration **(b) With reconfiguration**

Figure 38: Gantt chart for an example with two types of loads

More generally, we can establish the following theorem that compares the makespan with reconfiguration to the makespan without reconfiguration.

Theorem 5.3.1. *Let n be the total number of loads to be transported ($n = \sum_k n_k$). When $n \geq 2$, we have*

$$1 \leq \frac{T_W}{T_R} \leq \frac{n}{2}. \tag{5.7}$$

Proof. We distinguish two cases.

1. $T_R \geq 2$

   We have $T_W \leq n$ since we transport at least one load per period.

   $$1 \leq \frac{T_W}{T_R} \leq \frac{n}{T_R} \leq \frac{n}{2}. \tag{5.8}$$

2. $T_R = 1$

   Reconfiguration is not used and $T_W = T_R$. Since $n \geq 2$, it follows that $\frac{T_W}{T_R} = 1 \leq \frac{n}{2}$.

□

By generalizing the example of Figure 38, we can easily show that the upper-bound is tight. We consider n loads, $n_1 = n - 1$ loads of type 1, $n_2 = 1$ load of type 2, two configurations (1-bot and $(n-1)$-bot), c_{11}=1, $c_{12} = 0$, $c_{(n-1),1} = 1$, $c_{(n-1),2} = 1$ and a fleet of $N = n - 1$ bots. Then we have $T_R = 2$, $T_W = n$. Hence $T_W = \frac{n}{2} \cdot T_R$.

Conclusion

We have studied the problem of scheduling a fleet of reconfigurable robots with the objective to minimize the makespan. We have proposed a mathematical formulation of the problem which can be solved with a linear optimization solver. We have showed that reconfigurability can reduce drastically the makespan when boxes are located in different locations.

Conclusion and perspectives

The presented work considers the problem of load transportation by poly-robots in the context of intralogistics. The poly-robots in this dissertation are reconfigurables, that is they consist of elementary robots that can be assembled in different ways over time to adapt to the loads to be transported. Each poly-robot configuration has its own transportable load capacity and can be reconfigured after each transportation. The interest of the work is to find out how to determine the optimal number of elementary robots and which configurations is more profitable to use at a given time.

As shown in the study of the literature, over the past decade there has been a significant increase in the use of robots in warehouses, freeing operators from tedious tasks. While reconfigurable robots offer several advantages for warehouse transport operations. Firstly, the means of transport can be dynamically adapted to the load size or mass, which prevents to use oversized poly-robots and allows to re-affect the available elementary robots. Secondly, poly-robots can fit any warehouse, whatever the rack type, aisle width or door width. The elementary robots are interchangeable, which improves failure tolerance. In addition, a poly-robot has mechanical characteristics that make it more stable and maneuverable than a forklift. However, since this branch of robotics is only at the stage of its development, the question of the size of the fleet of reconfigurable robots has not been considered in the literature, despite its huge potential.

Our contributions on cooperative and reconfigurable models

First of all, in Chapter 2, a comparison of the fleet of robots without cooperation and with cooperation was made for homogeneous loads. In the first case, robots work alone and the minimum number of robots can be found using an analytical formula. In the

second case, the robot fleet consists of 1-bots and p-bots (only one possible configuration). The constructed mathematical model allows us to determine the most profitable configuration (number of robots that should cooperate) for the minimum cost of transportation. If the capacity of p elementary robots is smaller than the capacity of a single p-bot configuration, then using exclusively p-bot configurations or with a mix of 1-bots can lead to a significant cost decrease. Otherwise, it is optimal to use exclusively single robots. With an infinite horizon problem, that is a fleet of vehicles operating indefinitely, the models lead to simpler results.

In Chapter 3 we have added the ability to reconfigure the robots. Mathematical programs were obtained for load transportation with and without the possibility of reconfiguration. This allows us to find out which configuration of the robot in which period and for the transportation of which type of load we will use to minimize the transportation cost. Thanks to the programs, we can compare the cost of the fleet with and without reconfiguration. The reconfigurability can divide the minimum number of elementary robots up to a factor of K (with K the number of load types). For the special case of two types of loads and two configurations of robots, the gain in number of robots is limited by p (with p the number of elementary robots in the p configuration) but may be significant for small fleets. For the second special case, with unit capacities, the gain in number of robots is limited by $K-1$ (with K the number of load types). In a variant where the demand is per period of time and not on the whole time horizon, the gain in number of robots is not bounded.

Seeing a mathematical problem, the question of complexity is posed and considered in Chapter 4. The described problem of a reconfigurable robot fleet is strongly NP-Hard. In three special cases of single period, single load type or single configuration, the problem can be solved in polynomial time with appropriate dynamic programming algorithms. An efficient heuristic algorithm for the general case can successfully be applied even for large instances and has good performances on the tested instances.

Eventually, in Chapter 5, we also focused on the issue of minimizing transportation time. Our mathematical programs of the problem can compare the minimum transport time with and without the possibility of reconfiguration. The reconfigurability can reduce drastically the time to transport all loads from one place of the warehouse to another, when boxes are located in different locations.

Future research directions

This dissertation marked the beginning of a new branch of research in intralogistics by considering a fleet of reconfigurable robots. The models that we considered are naturally simplified compared to real situations. Several lines of research could be thus proposed by tackling more general problems, where some hypotheses of our work are weakened. As an example, an interesting question is the handling of traffic jams in the robot fleet. Another one would consist in examining of options when the same robot configuration can have several different transport capacities and when a p-bot can simultaneously transport several types of loads. Finally, it would be also of interest to consider stochastic processes such as robot breakdowns, demands and transportation.

We terminate this dissertation by focusing on two specific lines of research which sound challenging.

Different transportation times

We focus on a first extension of our initial model: having different locations for unloading areas. Each location could be at a different distance from the loading area, and accordingly has its own transportation time.

Initially, for each type of load, the transportation time lasted one period. We propose a new model where some loads need a larger transportation time, i.e. of several periods. This new characteristic is justified by at least two reasons. First, some loads might be heavy or voluminous and, hence, make the speed of the robot which carries it decrease. Second, the unloading area of some loads could be farther from the starting point. As a consequence, the travel time of the robot to transport these loads is increased. In fact, we propose an even more general model: the transportation time of some load can also depend on the configuration which carries it. For instance, some load could be transported in two periods with a 10-bot but in one period with a 4-bot - due to the fact that a 10-bot are heavier and more difficult to guide.

We remind that the initial entries of the model in Chapter 3 were parameters T, K, P, α, β, the number of loads of each type n_k and the capacities c_{pk}. We add a new entry τ_{pk} which is the number of periods of time needed to transport loads of type k by a p-bot.

A natural question is whether the techniques employed to tackle the initial problem could fit this more general version. Obviously, some variables have to be modified from the initial model. Indeed, when all transportation time used to be of one period and everything could be reconfigured at each step, variable N_{pk}^t denoted the number of p-bots carrying loads of type k during period t. It was sufficient to describe entirely the state of the warehouse. Now, we see that these p-bots, when $\tau_{pk} > 1$, can be at different step of their travel: either leaving the loading area at period t, being at the middle of their trip or arriving. For this reason, we fix a new variable $\tilde{N}_{pk}^t$ which corresponds to the number of p-bots leaving the loading area at period t in order to transport loads of type k. We fix $T_{pk} = T - \tau_{pk} + 1$ as the limit departure time for a p-bot with loads of type k. Hence, the total number of transported loads of type k is $\sum_{p=1}^{P} \sum_{t=1}^{T_{pk}} c_{pk} \cdot \tilde{N}_{pk}^t$. In summary, an ILP formulation of this problem, with variables N (total number of elementary robots) and $\tilde{N}_{pk}^t$, can be formulated.

$$\min \quad \alpha N + \beta \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{t=1}^{T_{pk}} p \cdot \tau_{pk} \cdot \tilde{N}_{pk}^t$$

subject to :

$$\sum_{p=1}^{P} \sum_{t=1}^{T_{pk}} c_{pk} \cdot \tilde{N}_{pk}^t \geq n_k \qquad \forall k$$

$$N \geq \sum_{k=1}^{K} \sum_{p=1}^{P} \sum_{s=[t-\tau_{pk}]^+ +1}^{t} \tilde{N}_{pk}^s \qquad t = 1, \cdots, T$$

$$N \in \mathbb{N},\ \tilde{N}_{pk}^t \in \mathbb{N} \qquad \forall k, \forall p, t = 1, \cdots, T$$

Complexity of the problem with 2 load types

Our last direction of research concerns the complexity of some specific cases of the reconfigurable model. In this dissertation, we showed that the **Multi_Bot** problem is strongly NP-complete. But, meanwhile, we identified exact polynomial-time algorithms for some cases, in particular when $K = 1$ (one type of load) but also when $T = 1$ (only one period) and $P = 1$ (one configuration). A natural question following our work is to determine where is the frontier between P and NP-hardness on the **Multi_Bot** problem.

Let us focus on one particular question of this context: what is the complexity of the **Multi_Bot** problem when $K = 2$? We thus consider a fleet of poly-robots which must

carry 2 types of loads. There are two ways to approach such question: trying either to design an exact polynomial-time algorithm for it or to build a reduction from some (strongly) NP-hard problem.

Our first idea is to consider the reduction presented in Theorem 4.2.4, page 73, from **Bin_Packing** to the general **Multi_Bot** problem. We wonder whether it could be adapted so that the case $K = 2$ could be proven strongly NP-hard.

By applying exactly this reduction, we prove in fact that the case $K = 2$ of the **Multi_Bot** problem is harder than **Bin_Packing_2**, which corresponds to **Bin_Packing** where the items admit only 2 distinct weights. Filippi and Agnetis [2005] showed that **Bin_Packing_2** is in P. Therefore, the reduction proposed in Theorem 4.2.4 does not allow us to prove that **Multi_Bot** with $K = 2$ is strongly NP-hard. Furthermore, it was proven recently [Goemans and Rothvoss, 2020] that **Bin_Packing** with a constant number of items (not only 2 but also 3, 4, 5,...) is in P. As a consequence, there is no hope that this reduction could help us to fix the complexity. On one hand, the observations made above imply that a trickier reduction would be needed if the case $K = 2$ is strongly NP-hard. On the other hand, they give us some hope to identify an exact polynomial-time algorithm. For the case $K = 1$, we proposed a dynamic programming algorithm (Section 4.3.3, page 78). A first approach could be to adapt this algorithm for the case $K = 2$.

The states of the DP algorithm proposed for $K = 1$ were made up of two components: value W, which represents the contribution of the p-bots with $p \neq p_0(k)$, and value EX which represents the contribution of the most efficient configurations $p_0(k)$. Thanks to Lemma 4.2.1, we know that keeping a similar value W in states is certainly a good idea as it will be polynomially bounded. Nevertheless, our analysis of value EX for the case $K = 1$ does not hold anymore now for $K = 2$. It seems that there is considerable effort to do in order to adapt this idea to the case $K = 2$.

In brief, starting from the DP algorithm for case $K = 1$ and modifying the state space in a suitable way would be our first approach to tackle the case $K = 2$.

Bibliography

R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot cooperation in the martha project. *IEEE Robotics & Automation Magazine*, 5(1):36–47, 1998.

Amazon Robotics. https://robotsguide.com/robots/kiva/, 2023.

M. Amjath, L. Kerbache, A. Elomri, and J. M. Smith. Fleet sizing of heterogeneous fleet of trucks in a material handling system using anylogic simulation modelling. *Proceedings of the 5th European International Conference on Industrial Engineering and Operations Management*, 2022.

T. Andersson. AGV & AMR ROBOTS. https://www.styleintelligence.com, 2022.

T. Arai, E. Pagello, L. E. Parker, et al. Advances in multi-robot systems. *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.

R. Arifin and P. J. Egbelu. Determination of vehicle requirements in automated guided vehicle systems: a statistical approach. *Production Planning & Control*, 11(3):258–270, 2000.

M. Asghari, S. M. J. M. Al-e, et al. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231:107899, 2021.

K. Azadeh, R. De Koster, and D. Roy. Robotized warehouse systems: Developments and research opportunities. *ERIM report series research in management Erasmus Research Institute of Management*, (ERS-2017-009-LIS), 2017.

I. Aziez, J.-F. Côté, and L. C. Coelho. Fleet sizing and routing of healthcare automated guided vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 161: 102679, 2022.

S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. A. Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue. *Journal of Field Robotics*, 24(11-12):943–967, 2007.

BALYO. https://www.balyo.com/agv-pallet-trucks/trucky, 2023.

P. Barosz, G. Gołda, and A. Kampa. Efficiency analysis of manufacturing line with industrial robots and human operators. *Applied Sciences*, 10(8):2862, 2020.

O. T. Baruwa and M. A. Piera. A coloured petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 54(16):4773–4792, 2016.

A. Baykasoğlu, K. Subulan, A. S. Taşan, and N. Dudaklı. A review of fleet planning problems in single and multimodal transportation systems. *Transportmetrica A: Transport Science*, 15(2):631–697, 2019.

G. J. Beaujon and M. A. Turnquist. A model for fleet sizing and vehicle allocation. *Transportation Science*, 25(1):19–45, 1991.

G. A. Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.

R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

BionicHIVE. https://www.bionichive.com/, 2023.

L. Bodin and B. Golden. Classification in vehicle routing and scheduling. *Networks*, 11 (2):97–108, 1981.

H. Bojinov, A. Casal, and T. Hogg. Emergent structures in modular self-reconfigurable robots. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1734–1741. IEEE, 2000.

D. Brandt, D. J. Christensen, and H. H. Lund. Atron robots: versatility from self-reconfigurable modules. In *2007 International Conference on Mechatronics and Automation*, pages 26–32. IEEE, 2007.

S. Broumi, A. Dey, M. Talea, A. Bakali, F. Smarandache, D. Nagarajan, M. Lathamaheswari, and R. Kumar. Shortest path problem using bellman algorithm under neutrosophic environment. *Complex & intelligent systems*, 5(4):409–416, 2019.

M. F. Campos and V. Kumar. Guilherme as pereira. *The International Journal of Robotics Research*, 23(7-8):783–795, 2004.

E. Cardarelli, V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi. Cooperative cloud robotics architecture for the coordination of multi-agv systems in industrial warehouses. *Mechatronics*, 45:1–13, 2017.

A. Castano, R. Chokkalingam, and P. Will. Autonomous and self-sufficient conro modules for reconfigurable robots. In *Distributed Autonomous Robotic Systems 4*, pages 155–164. Springer, 2000.

A. Caumond, P. Lacomme, A. Moukrim, and N. Tchernev. An MILP for scheduling problems in an FMS with one vehicle. *European Journal of Operational Research*, 199(3): 706–722, 2009.

M. Chaikovskaia, J.-P. Gayon, Z. E. Chebab, and J.-C. Fauroux. Sizing of a fleet of cooperative robots for the transport of homogeneous loads. In *2021 IEEE 17th International Conference on Automation Science and Engineering*, pages 1654–1659, 2021.

M. Chaikovskaia, J.-P. Gayon, and M. Marjollet. Sizing of a fleet of cooperative and reconfigurable robots for the transport of heterogeneous loads. In *2022 IEEE 18th International Conference on Automation Science and Engineering*, pages 2253–2258, 2022.

J. A. Chaves Osorio. *Design of a strategy to obtain safe paths from collaborative robot teamwork*. PhD thesis, Universidad Nacional de Colombia, 2021.

Z. E. Chebab. *Conception et commande collaborative de manipulateurs mobiles modulaires (C3M3)*. PhD thesis, 2018.

M.-S. Cheon, K. C. Furman, and D. Shaffer, Timothy. A modeling framework for railcar fleet sizing in the chemical industry. *Industrial & engineering chemistry research*, 51(29): 9825–9834, 2012.

G. Cormier and E. A. Gunn. A review of warehouse models. *European journal of operational research*, 58(1):3–13, 1992.

L. Deroussi, M. Gourgand, and N. Tchernev. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 46(8):2143–2164, 2008.

V. Digani. *Traffic Coordination for AGV Systems: an Ensemble Modeling Approach*. PhD thesis, 2016.

L. A. Dugatkin. *Cooperation among animals: an evolutionary perspective*. Oxford University Press on Demand, 1997.

P. J. Egbelu. The use of non-simulation approaches in estimating vehicle requirements in an automated guided based transport system. *Material flow*, 4(1-2):17–32, 1987.

T. Etezadi and J. Beasley. Vehicle fleet composition. *Journal of the Operational Research Society*, 34(1):87–91, 1983.

EXOTEC. https://www.exotec.com/system/robots/, 2023.

A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

C. Filippi and A. Agnetis. An asymptotically exact algorithm for the high-multiplicity bin packing problem. *Math. Program.*, 104(1):21–37, 2005.

D. B. M. Fontes and S. M. Homayouni. Joint production and transportation scheduling in flexible manufacturing systems. *Journal of Global Optimization*, 74(4):879–908, 2019.

V. Fourcassié, A. Dussutour, and J.-L. Deneubourg. Ant traffic rules. *Journal of Experimental Biology*, 213(14):2357–2363, 2010.

G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, 294(2):405–426, 2021.

Free3D. https://free3d.com/fr/3d-model/wood-pallet-set-2286.html, 2023.

T. Ganesharajah, N. G. Hall, and C. Sriskandarajah. Design and operational issues in AGV-served manufacturing systems. *Annals of operations Research*, 76(0):109–154, 1998.

M. R. Garey and D. S. Johnson. A guide to the theory of NP-completeness. *The Journal of Symbolic Logic*, 1979.

A. Gautam and S. Mohan. A review of research in multi-robot systems. In *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*, pages 1–5. IEEE, 2012.

C. M. Gifford. Review of selected mobile robot and robotic manipulator technologies. *Center of Remote Sensing of Ice Sheets, Technical Report, University of Kansas, Lawrence, KS*, 2006.

M. X. Goemans and T. Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6):38:1–38:21, 2020.

B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66, 1984.

Z. He, V. Aggarwal, and S. Y. Nof. Differentiated service policy in smart warehouse automation. *International Journal of Production Research*, 56(22):6956–6970, 2018.

B. Hichri, L. Adouane, J.-C. Fauroux, Y. Mezouar, and I. Doroftei. Cooperative mobile robot control architecture for lifting and transportation of any shape payload. In *Distributed Autonomous Robotic Systems*, pages 177–191. Springer, 2016.

A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, 37(12):2041–2061, 2010.

J. Hurink and S. Knust. Tabu search algorithms for job-shop problems with a single transport robot. *European journal of operational research*, 162(1):99–111, 2005.

INRS. https://www.inrs.fr/demarche/caces-certificat-aptitude-conduite-securite/ce-qu-il-faut-retenir.html, 2023.

J. Irawan, X. De Kestelier, N. Argyros, B. Lewis, and S. Gregson. A Reconfigurable Modular Swarm Robotic System for ISRU (In-Situ Resource Utilisation) Autonomous 3D Printing in Extreme Environments. In *Design Modelling Symposium Berlin*, pages 685–698. Springer, 2019.

ISO 445:2013. International organization for standardization 445:2013(en) pallets for materials handling — vocabulary. https://www.iso.org/obp/ui/#iso:std:iso:445:ed-4:v1:en, 2013.

ISO 6346:1995. Conteneurs pour le transport de marchandises — codage, identification et marquage. https://www.iso.org/fr/standard/20453.html, 1995.

M. R. Jahanshahi, W.-M. Shen, T. G. Mondal, M. Abdelbarr, S. F. Masri, and U. A. Qidwai. Reconfigurable swarm robots for structural health monitoring: a brief review. *International Journal of Intelligent Robotics and Applications*, 1:287–305, 2017.

D. Jung, G. Cheng, and A. Zelinsky. Experiments in realising cooperation between autonomous mobile robots. In *Experimental Robotics V*, pages 609–620. Springer, 1998.

Kardex. Vertical Carousel vs. Vertical Lift Modules. https://cdn.bfldr.com/EL3HU3A3/as/gz8cv83pk64837jbxr54hhz/BuyersGuide_US_VLMvsVCM?__hstc=210292298.792e451aaae1371b791700cc1b68fcc6.1634126841041.1634126841041.1634126841041.1&__hssc=210292298.7.1634126841041&__hsfp=581451351&hsCtaTracking=a529063e-ddc5-4925-b761-7c2cc93bd853%7Cc040a9ae-10a0-4445-b3cc-8b8fb2fdd12c, 2021a.

Kardex. Vertical vs. Horizontal Carousel Modules. https://cdn.bfldr.com/EL3HU3A3/as/t6455bgkqrcfkw5hhmw75zmq/BuyersGuide_US_VCMvHCM?__hstc=210292298.792e451aaae1371b791700cc1b68fcc6.1634126841041.1634126841041.1634126841041.1&__hssc=210292298.5.1634126841041&__hsfp=581451351&hsCtaTracking=81336701-45d7-4867-a0f3-93c649c7bd75%7Cfc738418-d82d-42ae-89c9-f9b6fab23a55, 2021b.

Kardex. https://www.kardex.com/en/, 2023.

Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte. The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological*, 70:239–254, 2014.

P.-H. Koo, J. Jang, and J. Suh. Estimation of part waiting time and fleet sizing in agv systems. *International journal of flexible Manufacturing Systems*, 16(3):211–228, 2004.

C. R. Kube and H. Zhang. Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218, 1993.

A. Kumar, D. Roy, V. Verter, and D. Sharma. Integrated fleet mix and routing decision for hazmat transportation: A developing country perspective. *European Journal of Operational Research*, 264(1):225–238, 2018.

P. Lacomme, M. Larabi, and N. Tchernev. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics*, 143(1):24–34, 2013.

V. T. Le et al. *Coopération dans les systèmes multi-robots: contribution au maintien de la connectivité et à l'allocation dynamique de rôles*. PhD thesis, Caen, 2010.

C. K. Lee, B. Lin, K. Ng, Y. Lv, and W. Tai. Smart robotic mobile fulfillment system with dynamic conflict-free strategies considering cyber-physical integration. *Advanced Engineering Informatics*, 42:100998, 2019.

H.-Y. Lee and C. C. Murray. Robotics in order picking: evaluating warehouse layouts for pick, place, and transport vehicle routing systems. *International Journal of Production Research*, 57(18):5821–5841, 2019.

N. Lenoble. *Optimisation de la préparation de commandes dans les entrepôts de distribution*. PhD thesis, Université Grenoble Alpes, 2017.

K. Lewandowski. Growth in the size of unit loads and shipping containers from antique to WWI. *Packaging Technology and Science*, 29(8-9):451–478, 2016.

X. Lyu, Y. Song, C. He, Q. Lei, and W. Guo. Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems. *IEEE Access*, 7:74909–74924, 2019.

MAGAZINO. https://www.magazino.eu/production-logistics/?lang=en, 2023.

N. Majcherczyk. *Communication Algorithms for Spatio-Temporal Cooperation in Multi-Robot Systems*. PhD thesis, 2020.

J. A. Manzolli, J. P. Trovão, and C. H. Antunes. A review of electric bus vehicles research topics–methods and trends. *Renewable and Sustainable Energy Reviews*, 159: 112211, 2022.

E. Mazareanu. Nombre total d'entrepôts aux États-unis 2007-2020. https://www.statista.com/statistics/873492/total-number-of-warehouses-united-states/, 2021.

MecaBotiX. https://www.mecabotix.com/, 2023.

Mecalux. Les avantages d'une gestion intelligente de l'entrepôt. https://www.mecalux.fr/articles-de-logistique/avantages-gestion-intelligente-entrepot, 2016.

M. Milenković and N. Bojović. A fuzzy random model for rail freight car fleet sizing problem. *Transportation Research Part C: Emerging Technologies*, 33:107–133, 2013.

M. Mohtasham, H. Mirzaei-Nasirabad, H. Askari-Nasab, and B. Alizadeh. Truck fleet size selection in open-pit mines based on the match factor using a minlp model. *Mining Technology*, 130(3):159–175, 2021.

M. C. Mountz, R. D'andrea, J. A. Laplante, I. D. P. Lyons, P. K. Mansfield, and B. W. Amsbury. Inventory system with mobile drive unit and inventory holder, July 22 2008. US Patent 7,402,018.

MouvBOX. https://mouvbox-france.com/palettes-container/, 2023.

A. Nath and R. Niyogi. A distributed approach for autonomous cooperative transportation. In *Robotics Software Design and Engineering*. IntechOpen, 2021.

F. R. Noreils. Cooperation between mobile robots and industrial applications: Some perspectives. *Proc. Application of Artificial Intelligence and Robotics to Nuclear Plants*, 1992.

E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund. Design of the atron lattice-based self-reconfigurable robot. *Autonomous Robots*, 21:165–183, 2006.

E. A. Oyekanlu, A. C. Smith, W. P. Thomas, G. Mulroy, D. Hitesh, M. Ramsey, D. J. Kuhn, J. D. Mcghinnis, S. C. Buonavita, N. A. Looper, et al. A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications. *IEEE Access*, 8:202312–202353, 2020.

G. Pantuso, K. Fagerholt, and L. M. Hvattum. A survey on maritime fleet size and mix problems. *European Journal of Operational Research*, 235(2):341–349, 2014.

M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*, volume 29. Springer, 2012.

M. A. Post, X.-T. Yan, and P. Letier. Modularity for the future in space robotics: A review. *Acta Astronautica*, 189:530–547, 2021.

A. Rahimi-Vahed, T. G. Crainic, M. Gendreau, and W. Rei. Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *Computers & Operations Research*, 53:9–23, 2015.

Y. Rizk, M. Awad, and E. W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.

A. Rjeb. *Dimensionnement d'une flotte de robots dans un entrepôt logistique*. PhD thesis, 2022.

A. Rjeb, J.-P. Gayon, and S. Norre. Sizing of a heterogeneous fleet of robots in a logistics warehouse. In *2021 IEEE 17th International Conference on Automation Science and Engineering*, pages 95–100, 2021a.

A. Rjeb, J.-P. Gayon, and S. Norre. Sizing of a homogeneous fleet of robots in a logistics warehouse: Transport operation between reception area and storage area. *IFAC-PapersOnLine*, 54(1):552–557, 2021b.

RoboCup. http://www.robocup.org/, 2021.

Roland Berger. Of robots and men – in logistics towards a confident vision of logistics in 2025. https://www.rolandberger.com/publications/publication_pdf/of_robots_and_men___in_logistics.pdf, 2016.

G. Romero, G. Durán, J. Marenco, and A. Weintraub. An approach for efficient ship routing. *International Transactions in Operational Research*, 20(6):767–794, 2013.

K. J. Roodbergen and I. F. Vis. A survey of literature on automated storage and retrieval systems. *European journal of operational research*, 194(2):343–362, 2009.

B. Rouwenhorst, B. Reuter, V. Stockrahm, G.-J. van Houtum, R. Mantel, and W. H. Zijm. Warehouse design and control: Framework and literature review. *European journal of operational research*, 122(3):515–533, 2000.

J. Sasaki, J. Ota, E. Yoshida, D. Kurabayashi, and T. Arai. Cooperating grasping of a large object by multiple mobile robots. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 1205–1210. IEEE, 1995.

J. Seo, J. Paik, and M. Yim. Modular reconfigurable robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:63–88, 2019.

M. Sha and R. Srinivasan. Fleet sizing in chemical supply chains using agent-based simulation. *Computers & Chemical Engineering*, 84:180–198, 2016.

W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh. Multimode locomotion via superbot reconfigurable robots. *Autonomous Robots*, 20:165–177, 2006.

D. Sinriech and J. Tanchoco. An economic model for determining agv fleet size. *International Journal of Production Research*, 30(6):1255–1268, 1992.

P. Sitek and J. Wikarek. Capacitated vehicle routing problem with pick-up and alternative delivery (cvrppad): model and implementation using hybrid approach. *Annals of Operations Research*, 273(1):257–277, 2019.

SSI Schaefer. https://www.ssi-schaefer.com/fr-fr, 2023.

K. Stoy, D. Brandt, D. J. Christensen, and D. Brandt. *Self-reconfigurable robots: an introduction*. Mit Press Cambridge, 2010.

G. Theraulaz, E. Bonabeau, S. C. Nicolis, R. V. Solé, V. Fourcassié, S. Blanco, R. Fournier, J.-L. Joly, P. Fernández, A. Grimal, et al. Spatial patterns in ant colonies. *Proceedings of the National Academy of Sciences*, 99(15):9645–9649, 2002.

E. Tuci, M. H. Alkilabi, and O. Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018.

A. Urru, M. Bonini, and W. Echelmeyer. Fleet-sizing of multi-load autonomous robots for material supply. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 244–249, 2018.

M. van Geest, B. Tekinerdogan, and C. Catal. Design of a reference architecture for developing smart warehouses in industry 4.0. *Computers in Industry*, 124:103343, 2021.

VecturaLogistique. https://www.vectura-logistique.fr/, 2021.

I. F. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006.

Y. Wang and C. W. De Silva. Multi-robot box-pushing: Single-agent q-learning vs. team q-learning. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3694–3699, 2006.

Z. Yan, N. Jouandeau, and A. Ali-Chérif. Multi-robot heuristic goods transportation. In *2012 6th IEEE International Conference Intelligent Systems*, pages 409–414, 2012.

Z. Yan, N. Jouandeau, and A. A. Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399, 2013.

Y.-J. Yao, Q.-H. Liu, X.-Y. Li, and L. Gao. A novel MILP model for job shop scheduling problem with mobile robots. *Robotics and Computer-Integrated Manufacturing*, 81: 102506, 2023.

M. Yim, D. G. Duff, and K. D. Roufas. Polybot: a modular reconfigurable robot. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 514–520, 2000.

J. Żak, A. Redmer, and P. Sawicki. Multiple objective optimization of the fleet sizing problem for road freight transportation. *Journal of advanced transportation*, 45(4):321–347, 2011.

B. Zou, X. Xu, R. De Koster, et al. Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. *European Journal of Operational Research*, 267(2): 733–753, 2018.