The crossdock truck scheduling problem : complexity, solving methods and uncertainty

Quentin Fabry

Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Quentin FABRY

Le 12 mai 2022

Ordonnancement de camions dans une plateforme logistique : complexité, méthodes de résolution et incertitudes

Ecole doctorale : **SYSTEMES**

Spécialité : **Génie Industriel et Informatique**

Unité de recherche :
LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par
Cyril BRIAND et Lotte BERGHMAN

Jury

M. Roel LEUS, Rapporteur
M. Jean-Charles BILLAUT, Rapporteur
Mme Anne-Laure LADIER, Examinatrice
Mme Elise VAREILLES, Examinatrice
M. Cyril BRIAND, Directeur de thèse
Mme Lotte BERGHMAN, Co-directrice de thèse

THE CROSSDOCK TRUCK SCHEDULING PROBLEM: COMPLEXITY, SOLVING METHODS AND UNCERTAINTY

QUENTIN FABRY

Quentin Fabry: *The crossdock truck scheduling problem: complexity, solving methods and uncertainty,* my water drop into the ocean of the research, December 2021

ABSTRACT

In recent years, the concept of crossdocking has received a lot of attention in literature. A crossdock is a logistics platform that promotes rapid product turnover through efficient synchronization of inbound and outbound trucks, with the volume of products stored kept as low as possible. Crossdocking raises many logistical problems, including the scheduling of incoming and outgoing trucks on the platform's doors. The classical objective considered in the literature for this problem is the minimization of the makespan, a very common criterion in scheduling. However, for crossdocking, minimizing the departure date of the last truck does not necessarily guarantee a good synchronization of the trucks. Therefore the makespan does not seem to be the most relevant objective.

In order to meet the need for synchronization and to favor fast rotations, our work proposes alternatively to minimize the total sojourn time of the pallets in the storage. We first study the deterministic version of this scheduling problem. Its complexity is detailed under different assumptions to better identify the elements leading to its NP-hardness. Afterwards, different solution methods are proposed: a classical integer linear programming method using time indexed decision variables, a critical-set algorithm that exploits a family of new valid inequalities with iterative addition of cuts, and methods based on constraint programming. A comparative analysis of these different methods is proposed.

In a second step, we study a non-deterministic version of our truck scheduling problem in which uncertainties on truck arrival dates are introduced in the form of equiprobable time intervals. A proactive-reactive scheduling method using the concept of groups of permutable tasks is proposed to cope with the uncertainties. Groups of permutable trucks are sequenced and assigned to the docks and later, during the scheduling implementation, a particular total truck order is chosen based on the actual realization of arrival dates, using a reactive algorithm.

RÉSUMÉ

La problématique dite de crossdocking a été source de beaucoup d'attention ces dernières années dans la littérature. Un crossdock est une plateforme logistique favorisant, par une synchronisation efficace des camions entrants et sortants, une rotation rapide des produits, le volume de produits stockés devant être le plus faible possible. Le crossdocking soulève de nombreux problèmes logistiques, dont notamment celui de l'ordonnancement des camions entrants et sortants sur les quais de la plateforme. L'objectif classiquement considéré dans la littérature pour ce problème est la minimisation du makespan, critère très répandu en ordonnancement. Pour le crossdocking néanmoins, minimiser la date de départ du dernier camion ne garantit pas nécessairement une bonne synchronisation des camions et le makespan ne semble donc pas être l'objectif le plus pertinent.

Pour répondre au besoin de synchronisation et favoriser les rotations rapides, notre travail propose de minimiser le temps de séjour total des palettes dans le stock. Nous étudions d'abord la version déterministe de ce problème d'ordonnancement. Sa complexité est détaillée selon différentes hypothèses pour identifier les éléments menant à sa NP-difficulté. Ensuite, différentes méthodes de résolutions sont proposées: une méthode classique de programmation linéaire en nombres entiers utilisant des variables de décision indexées par le temps, une formulation basée sur les ensembles critiques qui exploite une nouvelle famille d'inégalités valides avec ajout itératif de coupes et des méthodes basées sur la programmation par contraintes. Une analyse comparative de ces différentes méthodes est proposée.

Dans un second temps, nous étudions une version non-déterministe de notre problème d'ordonnancement dans laquelle des incertitudes sur les dates d'arrivées des camions sont introduites sous la forme d'intervalles de temps équiprobables. Une méthode d'ordonnancement proactive-réactive utilisant le concept de groupes d'opérations permutables est proposée pour faire face aux incertitudes. Des groupes de camions permutables sont séquencés et affectés aux quais puis, durant l'exécution d'ordonnancement, un ordre total est choisi en fonction de la réalisation des dates d'arrivées, à l'aide d'un algorithme réactif.

"Everything that lives is designed to end.
They are perpetually trapped in a never-ending spiral of life and death.
However, life is all about the struggle within this cycle."

Nier Automata - Pod 153

REMERCIEMENTS

Ce manuscrit, bien que signé d'un seul auteur, résulte de la participation de nombreuses personnes.

Tout naturellement, je remercie en premier lieu mes directeurs de thèse, Cyrille Briand et Lotte Berghman, qui, en plus de m'avoir fait confiance au début de ce projet, m'ont accompagné tout le long de ces trois ans. Je remercie également Alessandro Agnetis pour sa collaboration, qui fut grandemement appréciée. La mobilité initialement prévue à Sienne n'a pas pu avoir lieu avec regrets, mais les échanges par visio-conférence et le soutien dans les preuves théoriques furent d'une grande aide. Je remercie aussi Valentine Chassagne, stagiaire qui a pu participer au projet en cours, et qui a dû lire et comprendre une partie de mon code...

Sont aussi remerciés les membres de mon jury que je n'ai pas encore cité, J.C. Billaut, R. Leus, E. Vareilles et A.L. Ladier pour avoir pris le temps d'examiner et de juger mon travail. Ne sont pas oubliés le laboratoire LAAS-CNRS, qui m'a offert le cadre de travail ainsi que les membres de l'equipe ROC, permanents et doctorants que j'ai cotôyé pendant trois ans et qui m'ont aussi permis d'avancer sur la thèse, avec des conseils, des réponses, à la fois sur la théorie mais aussi sur la pratique (PLEX à installer, plateforme de calcul à utiliser,...). Ils ont aussi permis un appronfondissement de mes connaissances générales sur la RO lors de nombreuses discussions, ce qui est aussi une part essentielle de la thèse. Et ne sont pas négligés les aspects non-scientifiques, qui permettent aussi de mener à terme cette thèse, en particulier l'ambiance conviviale de l'équipe.

Dans cette même veine, je souhaite remercier ma famille, notamment mes parents qui malgré la distance, m'ont toujours soutenus dans mon parcours, et mon grand frère, qui m'a accompagné dans les projets de vie. Bien évidemment, je remercie tous mes amis, qui m'ont aidé à passer ces trois ans sainement, partageant boxe, bières-pizzas autour d'une table, whisky-cigare au bord de la piscine ou passes de rock dans les soirées privilégiées. Je nommerai au moins Maxime Bonnet, à qui j'adresse mes salutations doctorales, camarade de promo, qui a fait l'éxpérience de la thèse en même temps dans un autre laboratoire.

De manière un peu moins formelle, je voudrai adresser quelques mentions spéciales, liées aux évènements divers et variés que j'ai vé-

cus. Un grand merci au proprétaire de mon premier hébergement, qui m'a permis de loger facilement en arrivant à Toulouse, et pendant une bonne partie de la thèse, avant de changer pour un petit $22m^2$ bien cosy (que j'ai étrenné en plein confinement, sans box internet, car celle-ci m'attendait au relais-colis fleuriste, commerce qui s'avère non essentiel pendant la pandémie). Celà ne m'a pas empêché de télétravailler, Bouygues me fournissant des recharges de 100Go de 4G à la demande (j'en ai bien eu 4). Télétravail initiallement prévu sur la tour du laboratoire, que j'ai pu ramener chez moi pour l'occasion, qui ne s'est pas déroulé comme prévu car à partir du premier jour, le PC a décidé de s'éteindre aléatoirement. Ni la visite du premier technicien, pour remplacer la carte mère en parfaite état, ni la visite du deuxième pour remplacer l'alimentation défectueuse, par une autre alimentation défectueuse n'ont arrangé le problème. La solution a été le PC portable personnel, codant avec le sous-système linux pour windows, et github pour transférer le code à la plateforme de calcul. Cette adaptation m'a permis de passer le deuxième confinement dans une grande maison, chez mes parents. La thèse, avec un salaire et des horaires flexibles, aura aussi été l'occasion pour moi de passer le permis moto, une des meilleures décisions que j'ai pu prendre, un confort de vie pour se rendre au laboratoire (peu importe les bouchons) et un vrai plaisir à utiliser en balade. C'est une xsr 700 que j'ai choisie en première monture, et ce fût un vrai soutien. Pour finir ces remerciements, je dois mentionner qu'à la fin de rédaction de mon manuscrit, période un peu chargée où il faut écrire, et terminer un tas de choses, une double fracture de la clavicule m'a quelque peu ralenti, et je remercies encore pour tout le soutien qu'elles m'ont donné, les quelques personnes qui se reconnaitront. Ecrire au clavier à une main, passe encore, mais il devient vite compliqué de faire sa vaisselle, porter ses courses et un tas de mouvements anodins.

En resumé: *merci* à tous de m'avoir accompagné jusqu'au doctorat!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

With the globalization of markets, companies are expanding, carrying more products on larger distances. In order to maintain shipping cost competitive, the supply chain have to be adapted to the emergence of new needs. A specific supply chain organization must be developed in order to appropriately fulfill market expectations and customer needs. The supply chain is designed according to the distances to be travelled, the consolidation needs, the quantities to be shipped, the rotation frequency of the products, etc. The introduction of crossdocks as logistics nodes has been a revolution in the supply chain field in recent years. Acting like a logistic hub, the crossdock consolidates the product flows, with transshipment of goods from inbound to outbound trucks according to the destinations in order to have a quick transition and few storage. One of the first references on crossdocking was published in 1990 by Tsui and Chang (1990) who were interested in the problem of assigning trucks to the doors of a crossdock. Since then, thousands of articles have been written on a wide variety of crossdocking topics and the literature continues to grow significantly today.

Among all problems related to crossdocking, this work focuses on the crossdocking truck scheduling problem (CTSP), which was first addressed in (Chen et al., 2009). CTSP is often considered as the core of crossdocking. Indeed, it is crucial to ensure the synchronization of the trucks which is necessary for a fluid rotation of the products. Trucks cannot simply be loaded or unloaded when they enter the crossdock as there are only a limited number of doors available and limited storage space. There is a flow of pallets from incoming to outgoing trucks, and therefore the order in which the trucks are loaded and unloaded is important. In a traditional crossdock, doors are a limited resource and a truck should not stay at the door for too long. Most of the time, trucks have different arrival dates and must leave the platform on specific departure dates to deliver their products. The nature of the objective being considered can make the CTSP very difficult. Although makespan is often studied in the literature, its use is contested in the industry, and other more storage-oriented objectives may be preferred.

One main contribution of this thesis relies on the original storage-oriented objective that is studied. Indeed, in order to favor the synchronization of trucks, i.e. to reduce the storage over time, we focus on the minimization of the sojourn time of the products inside the storage area. The inspiration came from an industrial case working

with a carousel for consolidation and storage. Moreover, we choose a mixed mode to manage the doors, meaning that any door can handle both inbound and outbound trucks. Such a mode is known to allow a better synchronization, which is confirmed by both industrial feedback and scientific literature. The other assumptions we made are very common; i.e.: time windows associated with trucks, forbidden preemption, handover relationships.

Two main versions of the CTSP are studied in this thesis. In its first version, the problem is studied deterministically. We assume that the trucks arrive on time, and that they do not undergo a loading or unloading that is longer than expected (which is a bit unrealistic). Different solution approaches are considered, either based on mixed integer linear programming, constraint programming or more ad hoc branch-and-cut strategies. The second version of the CTSP studied involves uncertainties. This approach was developed following discussions with industrial users about their requirements, which confirmed that arrival delays of trucks are very frequent and often damaging to the synchronization. Decision makers are allowed to change the order of trucks on the dock doors, based on the actual arrival dates. We study how to guarantee the performance of a schedule that allows such flexibility.

This thesis is divided into three parts. The first part presents a detailed description of the problem, discusses our positioning, and provides complexity studies, taking into account various special cases of the CTSP. It also details how a set of realistic instances can been generated to assess our various solving approaches. It is structured into three chapters. Chapter 2 presents a review of related literature and justify the assumptions made in our work. The CTSP is formally stated in Chapter 3, in which useful definitions are introduced and complexity studies are made. The end of Part 1, Chapter 4, is dedicated to the generation of test instances.

The second part of this thesis focuses on the solving approaches. It offers five chapters. Chapter 5 details a basic time indexed formulation, developed as a reference to evaluate performances of our others formulations. Chapter 6 presents a very efficient quadratic formulation, only able to solve single-door CTSP instances. Chapter 7 introduces a branch-and-cut approach using the concept of a critical set of trucks and develops a new family of strong valid inequalities. Chapter 8 details several approaches to apply Constraint Programming on the CTSP. Chapter 9 finally compares the strengths and weaknesses of each formulation with computational experiments.

The last part is an attempt to address a non-deterministic version of the CTSP where uncertain arrival dates of trucks are considered. The concept of a group of permutable tasks is applied. Chapter 10 presents a review of the literature concerning robust crossdocking and recalls the original concept of a group of permutable tasks. Chap-

ter 11 shows how this concept can be applied on CTSP. Chapter 12 analyses the subproblems induced by the need to assess the quality of a flexible schedule, which is characterized by a group sequence of permutable trucks on each door of the crosssdock and also shows how these supbroblems can be solved. The generation of a group sequence is proposed.

Conclusions and future research perspectives are drawn in chapter 13. Please, note that the work presented in Chapter 3 concerning complexity issues has been presented at the 2020 PMS conference Fabry et al., 2021a and published in OR letters (Fabry et al., 2022). The work detailed in Chapter 12 concerning the evaluation of group sequences has been presented in APMS Fabry et al. (2021b).

Part I

CONTEXT AND PROBLEM DEFINITION

2

STATE OF THE ART

2.1 CONTEXT

2.1.0.1 *Emergence and importance of crossdocking*

With the increasing importance of specialization and internationalization, managing the supply chain becomes more and more challenging (Porter, 2008). Demand is difficult to predict and highly fluctuating, making goods turnover a priority. Moreover, modern society grants great importance to fast delivery. Amazon is a good example: with the ease of online shopping and fast deliveries, the e-commerce underwent an astonishing growth. Fast deliveries and the ability to manage a huge amount of products in little time, are now mandatory in the logistic chain and wasting time is more costly than ever, which is even more the case when we consider perishable products.

According to PWC, standard distribution centers with standard warehouses are not adapted to this mentality change (PwC Report, 2013). In classical storage, incoming products are sorted out and stored for some days on average. They are regrouped into full truckloads afterwards, depending on the needs of the customers. If a product level decreases below a predefined limit, a new order is placed. This practice is relatively easy to set up but has some important disadvantages that induce high logistic costs. In particular, a large area is needed for storage, numerous actions are needed to store and pick each product and the immobilization of the products delays their delivery and has a high set-up cost. In the last decades, a new approach to product consolidation / deconsolidation appears in research and industry. Crossdocking is a warehouse management concept in which items delivered to a warehouse by inbound trucks are immediately sorted out, reorganized based on customer demands and loaded into outbound trucks for delivery, avoiding excessive inventory at the warehouse (Apte and Viswanathan, 2000; Van Belle et al., 2012). Theoretically, a crossdock behaves like a logistic hub; but in practice, a minimum of storage is needed as items are held in storage for a short time period, generally less than 24 hours. Rarely, crossdocks can practice a zero-stock policy with direct transshipments only (Boysen, 2010), in case of refrigerated or frozen food for example.

The difficulty of coordinating inbound and outbound trucks in a crossdock comes from the restricted storage area and the limited sojourn time of the products in the warehouse (Boysen, 2010; Yu and J., 2008). Outbound trucks receive their loads from many inbound trucks and all trucks have to respect time availability constraints. When the

number of transshipment activities in a crossdock increases, the planning complexity increases as well. Up to hundreds of trucks (see Berghman et al. (2015) for an example in the automotive industry with 480 inbound trucks a day), each with its own availability in time, have to be allocated to a dock door and to a time slot, respecting the preceding and successive trucks and the available crossdock door capacity. The problem is complex to solve, and this difficulty slows down the crossdocking expansion.

In this thesis we address a problem called *Crossdock Truck Scheduling Problem* (CTSP), which consists in assigning a door to each truck and deciding the sequence in which trucks are processed at each door. CTSP concerns the tactical/operational level: trucks are scheduled and attributed to dock doors so as to minimize the storage usage during the transfer of the goods. The internal organization of the warehouse (scanning, sorting, transporting) is not explicitly taken into consideration, as we assume that the internal crossdock logistic capacity is sufficient to implement any arbitrary docking schedule.

Nowadays, crossdocking is a relatively well-known concept in the industry: more and more industrial and logistic companies develop crossdocking solutions. The Saddle Creek Report (Patterson, 2018) is often mentioned to show the importance of crossdocking in industry in recent years: a survey involving 219 respondents reports that 83.6% of logistic companies use or plan to use crossdocking. Indeed, the economic benefits provided by crossdocking are confirmed, as far as it is used for relevant products. Economic papers agree on the strength of crossdocking, whenever the necessary requirements, as constant demand rate on products, the density of the business etc., are present (Apte and Viswanathan, 2000; Waller et al., 2006; Galbreth et al., 2008). An empirical analysis strongly supports the success, and precises a methodology to identify products of interest from a case study (Duan et al., 2020). More than ever, delivering any product anywhere becomes a need, while production is more and more centralized. Crossdocks allow both consolidation and fast delivery, which is exactly the reason why it has such a major role in modern logistics. Walmart is a notable example of crossdock efficiency as its expansion and success is based on it (Stalk et al., 1992). We can mention UPS (Forger, 1995), Toyota (Witt, 1998), Dots (Napolitano, 2011), DHL (Fedtke and Boysen, 2017) among others.

Figure 1 illustrates the use and the interest of the crossdock, aiming to consolidate the product flow. Pallets from inbound trucks are spread into outbound trucks, with direct transshipment if it is possible, avoiding double handling and storage occupation, or with the help of temporary storage, in a centralized loading area. *LTL* stands for *Less Than Truckload*, commonly used to talk about trucks partially loaded, while *FTL* stands for *Full Truck Load* which is usually desired to transport products.

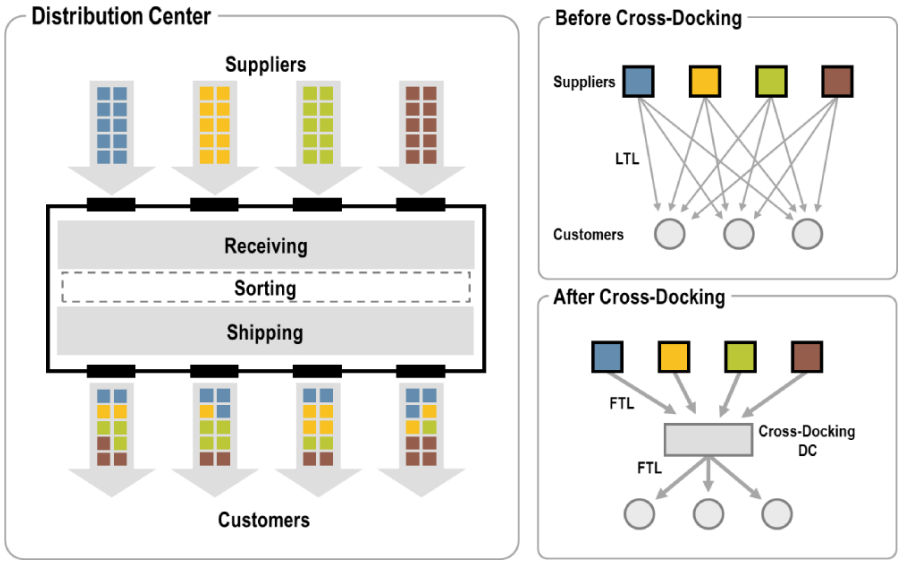

Figure 1: Illustration of the crossdock
Note: This image has been extensively plagiarized and appears on several commercial web sites without attribution

2.1.0.2 *Research and crossdocking*

Numerous researchers have focused on the crossdocking concept. However, it is worth noting that a gap between industry and research emerged, as highlighted by Ladier and Alpan (2016a). This paper relates a large growth of scientific papers about crossdocks. Several common assumptions are made in literature such as I-shape for the crossdock or no-preemption, which are relevant in practice. Concerning the service mode of the doors, literature mostly focuses on exclusive mode (doors are used exclusively for inbound or outbound), while the industry begins to use the mixed mode (doors can alternatively be assigned to inbound and outbound). Storage space and resource capacity are important constraints in the industry, but barely taken into account in the literature. Arrivals of trucks are often concentrated in industry, while this is not often mentioned in literature. Departure times are mostly constrained in reality, while half of the literature does not consider strict deadlines. Moreover, management of unplanned events and delays are very important in reality. The management of the workforce is also a critical point, although rarely taken into account in research. Finally, the main objective function in literature is the minimization of the makespan. However, this objective does not affect the synchronisation of the trucks nor the storage, although highly important in the crossdocking concept.

Optimization problems inside a crossdock are numerous, and the majors ones are covered by literature. Even if the concept of crossdocking is simple, its implementation is complex, as the synchronization between transporters raises different issues. Boysen and Fliedner

(2010) clearly identify and distinguish the many problems related to crossdocking, which are listed below:

- location of cross docking terminal(s);
- layout of the terminal;
- assignment of destinations to dock doors;
- vehicle routing;
- truck scheduling;
- resource scheduling inside the terminal;
- (un-)packing loads into (from) trucks.

These problems cannot be solved jointly due to the overall complexity. And, except some of them, they are not treated at the same decision level. Location and layout by example, are solved once, to build the crossdock, and condition the following problems. Figure 2 illustrates the various possible shapes of a crossdock, highly impacting the internal management. As an example Bartholdi and Gue (2004) discuss the shape efficiency to minimize the travel distance inside the crossdock, based on the number of doors. The travel distance inside the crossdock is used to measure this efficiency.

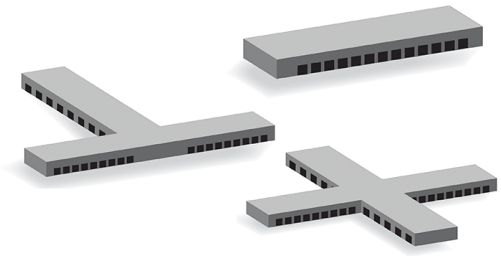

Figure 2: Example of different shapes for crossdock
Source: https://eurekapub.fr/productivite/2013/09/14/is-cross-docking-the-ultimate-panacea

Buijs et al. (2014) analyse the inter dependencies among these problems and particularly emphasize the need to consider the synchronization of local and network-wide crossdocking operations.

Following the classification of Boysen and Fliedner (2010), Van Belle et al. (2012) propose a valuable state-of-the-art, often cited in crossdocking papers. First, the authors define the crossdock, and contextualize the use of crossdocking as a strong tool in the supply chain to reduce costs, risks and storage space, to increase customer service and truck utilization and to improve delivery times. Nevertheless, some necessary conditions are listed, like low stock out costs and

stable demand. Afterwards, the authors enumerate characteristics of the crossdock, to precise the classification. They differentiate physical, operational and flow characteristics. The first category concerns the shape of the crossdock, the number of doors and the internal transportation system. Operational characteristics contain the service mode and whether or not preemption is allowed. The flow characteristics concern the arrival pattern of the trucks, the departure times, the product interchangeability, the temporary storage. Finally, several papers are presented according to the classification of Boysen and Fliedner (2010).

This thesis focuses on the truck scheduling problem and pays attention to ensure an efficient synchronisation of the trucks in crossdocks. Both the problems of assigning the doors to the trucks and scheduling the trucks on each door are considered together. It is worth noting that door assignment problems have been sometimes associated to scheduling problems in the literature.

In the truck-to-door standalone assignment problem, the spatial positioning of trucks is optimized to improve the internal management, the objective being to minimize the travelled distance of the pallets (Maknoon, 2013; Hermel et al., 2016). Maknoon and Baptiste (2009) work on the internal crossdocking management, to obtain the best assignment given a predetermined schedule. An assignment aiming only to minimize the travel distance of the pallets inside the distribution center may lead to a bad synchronisation of trucks. Zouhaier et al. (2016) detail and retrace the evolution of both problems in literature, as well as their mutual interactions. Rijal et al. (2019) defend the benefits of an integrated scheduling and assignment approach.

In this thesis, we are nevertheless focusing on the pure scheduling problem, and this already complex problem is mainly treated alone in literature.

2.2 LITERATURE ANALYSIS

2.2.1 *Variants of the crossdock truck scheduling problem in the literature*

As scheduling trucks on doors is central for proper synchronisation of crossdocks, the CTSP has been extensively studied in literature. In this state-of-the-art section, we will report various deterministic CTSP variants and solving approaches connected with our case. For a more extensive overview, the reader can refer to the very interesting state of the art proposed by Ladier and Alpan (2016a), discussing the relevance of current academic assumptions with usual industrial practices. Note that another literature analysis is proposed in Chapter 10 dedicated to literature dealing with crossdock scheduling under uncertainties.

Briskorn et al. (2021) recently considered the general scheduling problem of vehicles at transshipment terminals, seen as a synchronization problem consisting in the assignment of incoming vehicles to docking resources subject to handover relations. They define a relevant classification of the various problem variants and take interest in finding feasible door assignments. Handover relations model commodity exchanges between vehicles (a vehicle receiving a commodity cannot leave the system before the vehicle supplying it has arrived). The authors also introduce several parameters to classify the variants. They first consider that the set of doors is divided into groups, the doors of the same group sharing common characteristics required by the trucks: a truck is associated with one and only one group, given that the door assigned to it must belong to this group. Three kinds of handover relations are distinguished: i) in the "sym" case trucks exchange commodities (i.e., a commodity flow from truck u to truck v implies a flow from v to u and vice-versa); ii) in the "asym" case the flow is retrained to be unilateral; and iii) the "gen" case does not impose any flow restriction. The "inner" and "inter" parameters allow to distinguish between the situations where handover relations are set between truck associated with the same group or not. Preemption can be specified using status *"one"*, *"interrupt"*, and *"revisits"* where status *one* forbids preemption as the truck can only be docked once, status *interrupt* and *revisits* allow preemption, the truck being respectively restrained to stay or not at the same door. Two storage strategies are considered depending on whether the goods can be stored intermediately in the terminal ("sto" strategy) or not ("noSto" strategy). For instance, assuming two door groups, one refers to (one, sto, inter, asym) as the classical crossdock case with intermediate storage where two distinct sets of inbound and outbound trucks have to be unloaded and loaded, respectively, without any interruption.

Briskorn et al. (2021) do not consider release dates and deadlines, i.e., time windows associated to trucks. In addition, they assume that the trucks have a minimum handling time, which can be extended in the schedule if it becomes necessary to wait for other goods to arrive. Indeed, the handover relations between trucks imply that one outbound truck receiving pallets can leave only if all its connected inbound trucks have been docked. Any truck receiving and providing pallets is considered as both inbound and outbound. The authors concentrate their analysis on the feasibility of the handover relationships with respect to specific crossdock configurations. As we will see below, the assumptions taken in our work will be slightly different.

As mentioned by Ladier and Alpan (2016a), most articles in the literature do not allow preemption in CTSP because allowing it may have little benefit in practice, as the cost of interrupting and moving trucks is not negligible. In addition, due to the existence of due dates on truck departures, it may lead to infeasibility. However, preemption

has been studied in several papers, e.g., (Briskorn et al., 2021; Alpan et al., 2011; Larbi et al., 2011; Fazel Zarandi et al., 2016; Ye et al., 2018), which introduce objective functions that limit the caused time overhead.

The no-storage policy mentioned by Briskorn et al. (2021) is also rarely considered in practice, as it is commonly assumed that the storage capacity is sufficient to accommodate the flow of goods, especially to deal with the case where incoming and outgoing trucks are not physically present at the same time in the crossdock (i.e. their time window does not overlap). The relevance of the "Sto" policy is also justified in (Ladier and Alpan, 2016a) with respect to real-life experiences in crossdocks which tend to proof that a storage space, able to handle the unloading of trucks, is always suitable as it drastically enhances the fluidity of the flows. However, a very particular crossdock case with no-storage policies (*noSto*) is presented by Boysen and Fliedner (2010). This paper considers a full direct transshipment crossdock, which arises in the food industry, where strict cooling requirements forbid an intermediate storage inside the terminal. The authors took interest in minimizing the flow time, the processing time, and the tardiness of outbound trucks, using heuristic procedures.

The docking time, which may be variable when preemption is allowed, may also depend on the characteristics of the door. Nevertheless, the speed at which a truck is (un)loaded is mostly the same for all doors, see Golias et al. (2013) and Tadumadze et al. (2019) for the rare assumption that the handling time is dependent of the allocated door and sometimes even proportional to the number of goods (or pallets) to be moved, see e.g. (Guo et al., 2019). A few papers interestingly consider truck processing times by taking into account door efficiency, which may depend on workforce assignment, see e.g. (Carrera et al., 2008; Tadumadze et al., 2019).

The number of doors in the crossdock is also an important parameter that drastically influences both the efficiency of the transshipment in the crossdock and the complexity of the underlying problem of truck synchronization. In the literature, the number of open doors is generally constrained to be less than a given value, although this academic assumption does not really appear in the industry where the capacity of crossdocks is generally oversized to cope with possibly high truck rotation rates. Many papers consider very restricted cases with a single door or two doors (an inbound door to unload trucks and an outbound door for loading, see e.g, Monaco and Sammarra (2019)). In our work, the influence of the door capacity on the computational complexity of the problem will be studied considering a small number of doors. Nevertheless, the proposed solution methodologies will be able to cope with a high number of doors.

The doors can operate according to two different modes (as suggested by Bodnar et al. (2015)). A first option is to use the doors in ex-

clusive mode, i.e. each door is dedicated exclusively to unloading or loading operations. This mode tends to simplify the management of the product flow inside the crossdock, which becomes unidirectional (Boysen, 2010). However, some doors can operate in mixed mode if both inbound and outbound trucks are accepted, which guarantees better performance in terms of flow time, although the internal logistic within the crossdock becomes more complex, see e.g. Berghman et al. (2015). The mixed mode turns out to be more and more frequent in industry, as well as in the academic literature (Shakeri et al., 2012; Rijal et al., 2016; Hermel et al., 2016; Guo et al., 2019).

In (Berghman et al., 2015), a comparison between exclusive and mixed mode is presented. As expected, the authors show that with the same number of doors, better solutions are found in mixed mode. Moreover, they point out that only a few doors need to switch to mixed mode to benefit from the improvements. Such a flexible mode, efficiently dividing the doors into exclusive and mixed mode, is also discussed in (Rijal et al., 2016) and (Bodnar et al., 2015). It has the advantage of minimizing the perturbations on the internal logistics of the crossdock.

The consideration of arrival dates and/or departure due dates or deadlines for trucks is common in the literature, and almost systematic in the real world. The existence of time windows can make the CTSP infeasible and occasionally motivates the introduction of delay penalties, as discussed by Li et al. (2004). Their effect on the structure of the problem, as well as on the choice of the solving methods is important. For example, in the field of mixed integer linear programming, time-indexed formulations are known to be more appropriate than more classical position-indexed formulations when time-windows are strict (Monaco and Sammarra, 2019). In the literature, time windows are considered in various ways, e.g., as soft constraints (due dates), hard constraints (deadlines) or in a flexible way (Carrera et al., 2008).

Many articles consider earliness and tardiness penalties that reflect the just-in-time paradigm. Li et al. (2004), Alvarez-Pérez et al. (2009), Boloori Arabani et al. (2010), and Serrano et al. (2017) aim to minimize the earliness and the tardiness of the trucks. Meta-heuristics are used, with the exception of (Serrano et al., 2017) using solution method based on industrial solver. Ladier and Alpan (2018), Ladier and Alpan (2013) and Bodnar et al. (2015) consider two objectives: while minimizing the earliness and/or the tardiness, they also minimize storage. The resolution of the normal scale problem is done with heuristics or metaheuristics. Boloori Arabani et al. (2011a) and Boloori Arabani et al. (2012) consider the lateness with the usual makespan objective and use metaheuristics. Rijal et al. (2016) and Assadi and Bagheri (2016) and Rijal et al. (2019) want to solve the truck to door assignment problem as a scheduling problem. The objective is to min-

imize the lateness and the travel cost of pallets inside the crossdock. Moreover, Rijal et al. (2016) and Rijal et al. (2019) consider a third objective, respectively minimizing storage and minimizing makespan. Heuristics or metaheuristics are provided to solve the problem. Golias et al. (2013) and Fazel Zarandi et al. (2016) have less common objectives. The first ones minimize the service cost induced by lateness, and the second ones minimize earliness and tardiness while limiting preemption.

Many authors, e.g. Boysen and Fliedner (2010) and Berghman et al. (2015), consider strict time windows using deadlines. For example, Diglio et al. (2017) aim to minimize the sum of the outbound truck completion times, which is quite unusual under time window constraints. Tadumadze et al. (2019) focus on inbound trucks (with the schedule of outbound trucks assumed to be fixed): the authors optimize the allocation of workers at the doors to speed up the unloading process. Note that other papers consider only release times, e.g. (Serrano et al., 2017) or only deadlines, e.g. (Boloori Arabani et al., 2011b), that reflect the arrival time and departure time of trucks, respectively. In (Yu and J., 2008), a special case is treated where some trucks are time constrained while others are free.

Another feature concerns the interchangeability of pallets loaded into the outbound trucks. For instance in (Briskorn et al., 2013), as pallets of identical products are considered, they can be loaded into various trucks, without any restriction. In that case, there are no precedence constraints between inbound and outbound trucks, but only inventory constraints expressing the requirement for each outbound truck. In (Ladier, 2015), a more general case is described where, since any pallet is assigned to a specific customer, it can be assigned to any truck serving this customer.

2.2.2 *Focus on the objective functions*

The various objectives considered in the literature for the CTSP have to be reported. The minimization of the makespan is very common, e.g. (Van Belle et al., 2012), (Yu and J., 2008), and (Gaudioso et al., 2020)). Nevertheless, the relevance of such a criterion in the crossdocking context has been reported to be questionable by Hedler-Staudt et al. (2015) as, minimizing the duration of the schedule does not really reflect the need of having a good truck synchronization; i.e. a smooth flow of goods and a low retention level in the storage. This issue is also raised in (Ladier and Alpan, 2016a) where the authors observe the over-presence of the makespan objective in the review of literature, while this objective is not considered as important by most of the industrials participating in their study. In a more recent paper, Ladier and Alpan (2018) gave privilege to more storage-oriented objectives. Alpan et al. (2011) and Larbi et al. (2011) aim to minimize the storage

assuming that a handling cost has to be paid when a pallet cannot be directly transferred from one truck to another. They combine this objective with the one of minimizing the number of truck replacements during docking (as they allow preemption). Sadykov (2012) also aims to minimize the handling cost without time windows and only two exclusive doors. Once again, trucks are sequenced in such a way that direct transshipment is favored. In her PhD thesis, Ladier (2015) also considers the assumption of Alpan et al. (2011), but without preemption: a fixed cost is assigned to any pallet placed into the storage. Her objective function aggregates the criterion of minimizing the total handling costs with the one of minimizing the tardiness of trucks (truck departure times are seen as due dates).

Bodnar et al. (2015) also consider the minimization of the handling costs and truck tardiness with dedicated inbound / outbound doors and a limited number of mixed doors. In this paper, they assume handover relationships (i.e., pallets are pre-assigned to trucks), no-preemption and unitary processing times for inbound trucks. They propose an adaptive large neighborhood search heuristic to solve the considered problem. Guo et al. (2019) consider a direct zero-stock transshipment model such that loads are directly exchanged without intermediate storage or double handling. They propose MILP formulations, capable of minimizing the total weighted sum of realized transfer times, as well as an efficient heuristic. Briskorn et al. (2021) also study the direct transshipment model by considering that the docking times of outbound trucks are not fixed: an outbound truck can be docked before the inbound trucks are unloaded and wait for missing pallets. As seen above, it is desirable to take into account the storage performance within the objective function. Note the importance of both the number of products transiting in the stock as well as their sojourn time. Indeed, with respect to the turnover performance, it may be preferable to have a reasonably large quantity of products in stock if their sojourn time is very short, rather than a small quantity whose sojourn time would be very long. This observation led Berghman et al. (2015) to focus on the minimization of the total sojourn time of the pallets: a time-indexed MILP formulation is proposed to solve this version of CTSP with time windows.

The sojourn time is particularly important when the storage system has been designed to achieve the just-in-time logic. As an illustration, Figure 3 describes a system in the food industry in which a conveyor belt carousel includes input stations (i.e., inbound doors) at which products are introduced and output stations (i.e., outbound doors) to which products are transferred according to the trucks assigned to them. Since the capacity of the conveyor is limited, the goal here is to ensure that a product accepted on the conveyor is transferred as quickly as possible to avoid congestion.

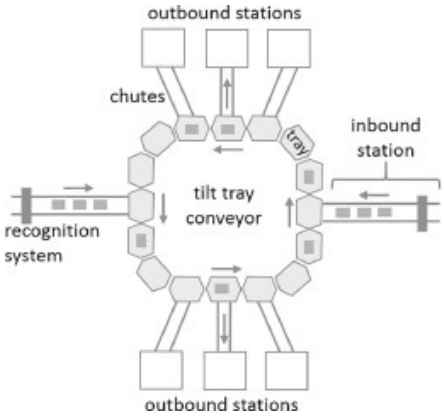

Figure 3: Example of carousel conveyor
(Boysen et al., 2019)

2.2.3 *Working assumptions*

In line with Berghman et al. (2015) and the previous example, our work will focus on the total time spent in the system by the goods (i.e., the total sojourn time). A good will be considered physically present in the inventory from when the truck carrying it is docked, until the truck picking it up is docked in its turn. Therefore, if these two trucks are synchronized (i.e. placed at the door at the same time), the storage time is zero.

For the doors, we assume a mixed-mode which offers more flexibility, as often pointed out in the literature. It will also be assumed that pallet flows have been defined in advance in the crossdock (i.e., pallet interchangeability is not allowed) and that loading or unloading of trucks cannot be interrupted. Therefore, we will assume transfer relationships between trucks, each transfer relationship being associated with an explicit pallet flow and a start-start precedence constraint. For that reason, an outbound truck can only be docked if all the inbound trucks feeding it are already docked or have been previously unloaded.

It appears from the literature that the influence of time windows is crucial in crossdocking, both in terms of computational complexity and logistic performance. Therefore, we decided to study, for each CTSP instance, different time window scenarios, starting with strict time windows and then gradually relaxing them until, at the maximum extension, time windows can be ignored.

The processing times for unloading or loading trucks reflect the ability of internal logistics to efficiently handle pallet traffic, i.e. the various handling operations. As several authors have done in the literature, we will assume that the handling facilities are sized to handle any rate of truck traffic. In our work, we decide to explore two scenarios. In the former, the handling time will be assumed to be proportional to the number of pallets, which seems to be a relevant

assumption with respect to reality. The previous assumption gives a particular structure to the problem as there is a relationship between the pallet flows and the handling times, which possibly makes the problem simpler. The second scenario therefore assumes independent processing times.

In this work, the focus is essentially on the resolution of the CTSP by means of exact methods. The ambition is to tackle instances of realistic size (with respect to industrial practices) and to find, if possible, optimal solutions or, at least, feasible solutions with a lower bound as performance guarantee. Thus, the development of heuristics or metaheuristics is not addressed, even if their design remains an important area of research and such methods are certainly very useful from a practical point of view.

3

PROBLEM STATEMENT AND COMPLEXITY

In this chapter, we develop the problem and and its statement. Its computational complexity is also studied with a focus on some special cases.

3.1 PROBLEM STATEMENT

3.1.1 *Elements of the problem*

Given a crossdocking warehouse with n doors, a set I of *inbound trucks* carries goods to be unloaded, sorted and loaded on a set O of *outbound trucks*. We let $U = I \cup O$ be the set of all trucks, and $k = |U|$ denotes the total number of trucks. In the following, we will usually use u to refer to any truck in U, meaning it can be inbound or outbound. Similarly, we will use i to refer to a truck in I and o to refer to a truck in O. Let w_{io} denote the number of *pallets* to be transferred from truck i to truck o. Loading/unloading operations take place once the trucks are docked at one of the n doors of the warehouse. They cannot be preempted. Each door can be used both for loading and unloading. The unloading and loading times of trucks $i \in I$ and $o \in O$ are referred to as p_i and p_o, respectively. We let $w_i = \sum_o w_{io}$ and $w_o = \sum_i w_{io}$ denote the total number of pallets to be unloaded from i and, respectively, the total number of pallets to be loaded on o.

Handover relations among trucks are represented by the handover graph $G_H = (I, O, A)$. In G_H, the two node sets correspond to inbound and outbound trucks, respectively, and there is an arc $(i, o) \in A$ whenever $w_{io} > 0$. Letting s_i and s_o denote the time at which truck $i \in I$ starts being unloaded and, respectively, the time at which truck $o \in O$ starts being loaded, we require that if $(i, o) \in A$, in a feasible schedule $s_i \leqslant s_o$, i.e., the handover graph G_H specifies start-start precedence constraints. This means that an outbound truck o can be docked – and hence, it can start being loaded – only when all trucks i with $(i, o) \in A$ have been docked. Although this may appear as a conservative assumption, we note that it avoids modeling the detailed movement of individual pallets between each truck pair and their unload/load schedule, which would lead to a considerably more complex model. Moreover, if all the pallets required by an outbound truck o are present in the terminal when o is docked, then its loading can take place *non preemptively* in an interval of length p_o. As we do not allow docking a truck more than once, in this way we are

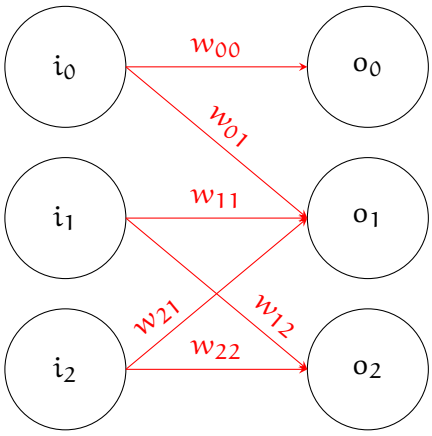

Figure 4: Example of a transfer graph.

sure that each truck (either inbound or outbound) is docked exactly for the time strictly needed to load/unload it. On the contrary, if we let an outbound truck o be docked even before all corresponding inbound trucks have arrived, the time spent by o at the door would in general depend on the other trucks' schedule, and attention should be paid to avoid deadlocks. Moreover, since we consider terminals allowing direct transshipment of goods (e.g. (Guo et al., 2019)), start-start precedence constraints appear more appropriate than the more classical (and common) end-start precedence constraints. If we required that all the inbound trucks i feeding an outbound truck o be completely unloaded before o starts being loaded, even when both trucks i and o such that $(i, o) \in A$ are present, pallets should necessarily spend some time in intermediate storage, which we want to avoid as far as possible. (In any case, note that if $n = 1$, the distinction between start-start and end-start precedences disappears.)

Finally, we refer to r_u and d_u as the release date and deadline of truck $u \in U$, $[r_u, d_u]$ being the time window during which the truck is present in the crossdock. The *Crossdocking Truck Scheduling Problem* (CTSP) consists in determining processing start times s_i and s_o for all $i \in I$ and $o \in O$, so as to minimize the *total sojourn time*, i.e., the total time spent in the warehouse by all pallets.

Figure 4 gives a small example of a handover graph. There are three inbound and three outbound trucks. The pallets of the inbound truck i_0 are loaded on the outbound trucks o_0 and o_1, while the pallets carried by both inbound trucks i_1 and i_2 are loaded on both the outbound trucks o_1 and o_2.

As we assume that, given a number of doors, there is sufficient workforce to (un)load all currently docked trucks at the same time, a truck assigned to a door never waits for the availability of a material handler. Pallets can be transshipped directly from an inbound to an outbound truck if both trucks are simultaneously docked. Otherwise, pallets are temporarily stored to be loaded later.

In this thesis, we consider two distinct cases concerning the processing times of the trucks.

(i) In the *unrelated* case, no relationship exists between the number of pallets that need to be loaded/unloaded and the processing time of a truck (i.e., in general, for any two trucks u and $v \in U$, one may have $w_u/p_u \neq w_v/p_v$);

(ii) In the *correlated* case, the processing time of a truck is proportional to the number of pallets that must be loaded/unloaded, i.e. $w_u \propto p_u$, $u \in U$.

3.1.2 *Objective*

For the objective function, we introduced the idea to reduce the total sojourn time in order to minimize the storage, speed up the turnover of goods, and promote direct transshipment. The sojourn time of one pallet transferred from truck i to truck o equals $s_o - s_i$, so the total sojourn time equals:

$$\sum_{(i,o)\in I\times O} w_{io}(s_o - s_i). \tag{1}$$

CORRELATED CASE: In the correlated case, we can rewrite the objective function. If we define one time unit as the time required to handle one pallet, processing times can be expressed in terms of number of pallets to be moved. In that case, it holds that:

$$p_i = w_i = \sum_{o\in O} w_{io} \tag{2}$$

$$p_o = w_o = \sum_{i\in I} w_{io} \tag{3}$$

$$\sum_{o\in O} p_o = \sum_{i\in I} p_i \tag{4}$$

As the total load of all inbound trucks is equal to the total load of all outbound trucks $\sum_{o\in O} w_o = \sum_{i\in I} w_i$ (pallets conservation), the total processing time of loading are equal to the total processing time of unloading, as specified in equation (4). Due to equations (2) and (3), it is easy to show that solving the problem in the *correlated* case is equivalent to finding a feasible schedule that minimizes

$$\sum_{o\in O} p_o s_o - \sum_{i\in I} p_i s_i. \tag{5}$$

3.1.3 *Constraints*

The constraints of the model are specified in the following. As said before, we do not consider workforce as we assumed that it is sufficient for the number of doors. If the workforce is weaker, one need

to consider the number of doors handled by the workforce. In that way, a truck assigned to a door never waits for the availability of a material handler. We also assume that loading and unloading tasks cannot be preempted.

In both the correlated and the unrelated case, the defined objective function is subject to the following constraints:

$$s_o - s_i \geqslant 0 \qquad \forall (i,o) \tag{6}$$

$$s_u \geqslant r_u \qquad \forall u \in U \tag{7}$$

$$s_u \leqslant d_u - p_u \qquad \forall u \in U \tag{8}$$

$$|U_\tau| \leqslant n \qquad \forall \tau \in \mathcal{T} \tag{9}$$

with $U_\tau = \{u \in U \,|\, s_u < \tau \leqslant s_u + p_u\}$ the set containing all tasks being executed during time period τ and $\mathcal{T}$ the set containing all time periods considered (time horizon).

Constraints (6) impose start-start precedences between inbound and outbound trucks sharing a pallet flow. Constraints (7) and (8) allow each truck to be (un)loaded during its available time window only. Finally, Constraints (9) model the capacity and ensure that the number of trucks simultaneously docked never exceeds the number of available doors.

We refer to this problem as the Crossdocking Truck Scheduling Problem (CTSP). In the sequel of this thesis we denote by CTSP(n,U) the problem with n doors and unrelated processing times. Similarly, CTSP(n,C) refers to the problem with n doors and correlated processing times.

In what follows we establish the complexity of various special cases of CTSP.

3.1.4 *Illustration*

We introduce an example in Figure 5. This example introduces an instance with 2 inbound trucks and 3 outbound trucks. The handover graph is presented on the left, with inbound trucks in blue and outbound trucks in red. The number of pallets carried by each inbound truck are indicated on the left, and the number of pallets carried by each outbound truck are indicated on the right. The exact number of pallets unloaded from each specific inbound truck and loaded to each specific outbound truck are specified in the middle. The crossdock has 2 doors. On the right side of the figure, the time windows of all trucks are detailed. In this example, we consider a correlated case, with load proportional to the time of loading/unloading. Even, the time scale is defined as one time unit correspond to the time needed to load/unload a pallet. Truck 1, as an example, needs 5 time units to be unloaded.

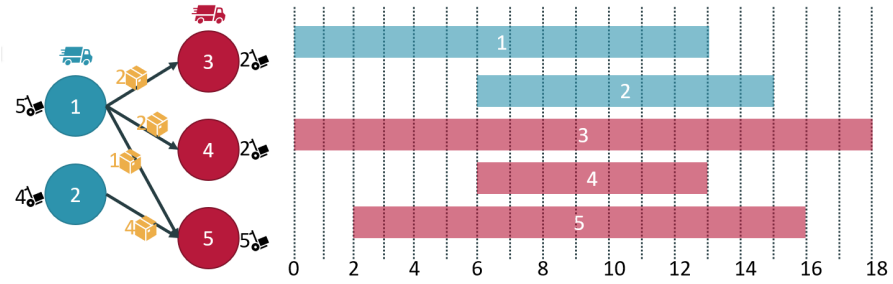

Figure 5: Illustration of a CSTP instance

Figure 6 details an optimal solution to the introduced instance. This solution respects all the constraints. Truck 1 starts before or parallel to all the outbound trucks (all connected to 1). Truck 2 starts before or parallel to truck 5 (the only connected truck). Moreover, during the entire schedule, there are no more than 2 trucks docked at the same time. We can observe that this optimal solution proposes direct transshipment with parallel i/o-trucks.

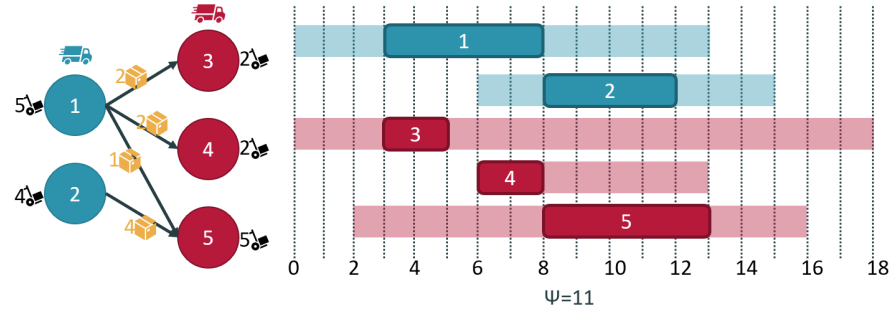

Figure 6: Illustration of a CSTP optimal solution

3.2 COMPLEXITY RESULTS

For general time windows, even with a single door, CTSP is obviously strongly NP-hard, as finding a feasible schedule is equivalent to finding a feasible solution to an instance of the single machine scheduling problem with time windows, which is NP-Complete (Pinedo, 2012).

In this section, we focus on the complexity of CTSP when there are no binding time windows. In particular, we study how the number of doors, the structure of the handover graph and unrelated/correlated processing times affect the problem complexity. We particularly ex-

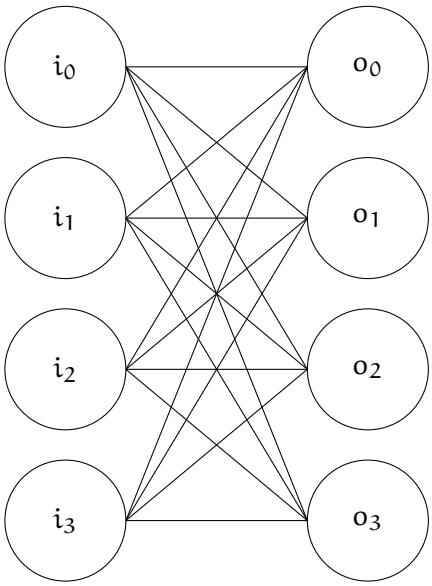

Figure 7: An instance of CTSP(1,Unr) in which G_H is a 1-biclique.

hibit some polynomial single-door CTSP cases that become NP-hard once an additional door is considered.

3.2.1 *Single-door problem*

In this section, we consider CTSP(1,·), i.e., the crossdock with a single door ($n = 1$). We first address a polynomial special case and then establish the complexity of CTSP(1,Unr).

3.2.1.1 *Bicliques, correlated and unrelated*

A *biclique* is a complete bipartite graph. In this section we address the case in which the handover graph G_H is a *h-biclique*, i.e., a collection of h disjoint bicliques.This means that I and O are partitioned into h subsets $I_1, \ldots, I_h$ and $O_1, \ldots, O_h$ respectively, such that $w_{io} > 0$ for any $(i, o) \in I_j \times O_j$, $j = 1, \ldots, h$. (See Figure 7 for an example with $h = 1$ and Figure 8 for an example with $h = 3$.)

Theorem 1. *If G_H is a 1-biclique, CTSP(1,Unr) is solved by first sequencing all inbound trucks in non-increasing order of the ratio p_i/w_i, then all outbound trucks in non-decreasing order of the ratio p_o/w_o.*

Proof. If G_H is a 1-biclique, in any feasible schedule all inbound trucks must be consecutively sequenced on the door before all outbound trucks. So, the problem consists in deciding the order within inbound and outbound trucks. Let us first consider inbound trucks. Given any feasible schedule σ, let i and j be any two consecutively scheduled inbound trucks, and σ' the schedule obtained by swapping i and j. Let $P(o)$ denote the total processing time of the trucks scheduled

after j in σ and before the outbound truck o. The contribution of i and j to the value of the objective function in σ is:

$$\sum_{o \in O} w_{io} \left(p_i + p_j + P(o)\right) + \sum_{o \in O} w_{jo} \left(p_j + P(o)\right),$$

While in σ', the contribution of i and j is

$$\sum_{o \in O} w_{jo} \left(p_j + p_i + P(o)\right) + \sum_{o \in O} w_{io} \left(p_i + P(o)\right),$$

hence, $f(\sigma) \geqslant f(\sigma')$ if and only if:

$$\sum_{o \in O} w_{io}(p_i + p_j) + \sum_{o \in O} w_{jo} p_j \geqslant \sum_{o \in O} w_{jo}(p_i + p_j) + \sum_{o \in O} w_{io} p_i,$$

i.e.

$$\frac{p_j}{w_j} \geqslant \frac{p_i}{w_i}.$$

Consequently, it is always profitable to schedule the inbound trucks by non-increasing values of ratios p_i/w_i. A symmetrical argument applied to outbound trucks shows that the outbound trucks should be sequenced by non-decreasing values of the ratios p_o/w_o. □

When considering CTSP(1,Cor), equations (2) and (3) imply the following.

Corollary 1.1. *Given an instance of CTSP(1,Cor), if G_H is a 1-biclique, the problem is solved by any schedule in which all inbound trucks are scheduled in any order before all outbound trucks, in any order.*

Proof. The theorem follows from the fact that in CTSP(1,Cor) equations (2) and (3) hold. □

Theorem 1 easily extends to the case where G_H is a h-biclique (like in Figure 8), as it can be easily checked that there is no gain in interleaving trucks belonging to different bicliques.

https://www.overleaf.com/project/5fc4c2821faca54f6f73bfod

Corollary 1.2. *When G_H is a h-biclique, CTSP(1,Unr) is solved by consecutively sequencing the trucks involved in the same biclique, as dictated by Theorem 1, and sequencing the bicliques in any order.*

Proof. It is never profitable to interleave trucks from different bicliques, as this increases total sojourn time, i.e., after unloading the first truck of a biclique, the best decision is always to unload all the inbound truck of that biclique and then to load all the outbound trucks of that biclique. The order in which the bicliques are processed is immaterial. □

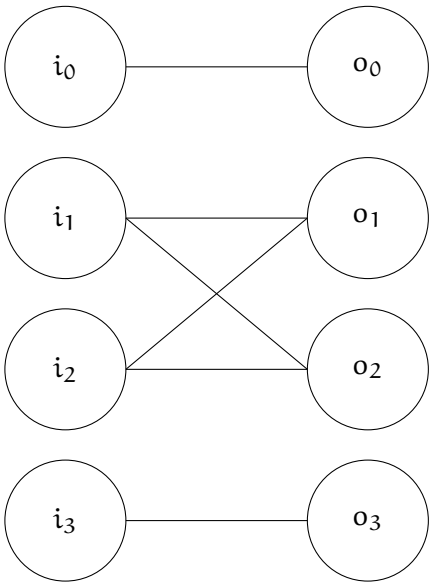

Figure 8: An instance of CTSP(1,Unr) when G_H is a 3-biclique.

3.2.1.2 *CTSP(1,Unr) with general* G_H

Let us now consider problem CTSP(1,Unr) where G_H is a general bipartite graph. The decision version of this problem is considered below.

"Given a set I *of inbound trucks and a set* O *of outbound trucks (with unloading and loading times* p_i *and* p_o *respectively), handover graph* $G_H = (I, O, A)$*, a pallet flow amount* w_{io} *for each* $(i, o) \in A$*, and a positive integer* H*, is there a truck sequence at the door such that the total sojourn time does not exceed* H*?"*

To prove the complexity of CTSP(1,Unr), let us recall the well-known strongly NP-complete problem OPTIMAL LINEAR ARRANGEMENT (OLA) (Garey and Johnson, 1978):

"Given an undirected graph $G = (V, E)$*, with* $v = |V|$ *nodes and* $m = |E|$ *edges, and a positive integer* K*, is there a numbering* $f(i)$ *of the nodes such that* $\sum_{\{i,j\}\in E} |f(i) - f(j)| \leqslant K$*?"*

Theorem 2. *CTSP(1,Unr) is strongly NP-complete.*

Proof. Obvioulsy, CTSP(1, Unr) is in NP. Given an instance of OLA (see Figure 10(a)), let us define an instance of CTSP(1,Unr) as follows (see Figure 10(b)).

- There are v inbound trucks, each corresponding to a node of G, and therefore called *node-trucks*;
- There are m outbound trucks, each corresponding to an edge $\{i, j\}$ of G and called *edge-trucks*;

- Each edge-truck $o = \{i, j\}$ receives exactly 1 pallet from node-truck i and 1 from node-truck j (so, for $o = \{i, j\}$, $w_{io} = 1$ and $w_{jo} = 1$).
- For all $i \in I$, $p_i = M$ while for all $o \in O$, $p_o = 1$, where M is a sufficiently large integer, say $M \geqslant 2m^2$.

We want to show that a truck schedule with a total time spent by the pallets in the warehouse smaller than or equal to $H = KM + 2mM + 2m^2$ exists if and only if an OLA with value not exceeding K exists.

Lets consider any truck schedule σ. Every edge-truck $\{i, j\}$ is scheduled after the last of both node-trucks i and j and before any other node-truck is scheduled. In fact, consider two node-trucks i and j, and an edge-truck $\{i, j\}$, and suppose that between the completion of the unloading of j and the start of loading of $\{i, j\}$, another node-truck, say u, is unloaded (see Figure 9(a)). Then, moving the loading of $\{i, j\}$ before the unloading of u decreases by M the sojourn time of the two pallets loaded on truck $\{i, j\}$, and indeed decreases also the sojourn time of all the pallets transported by u, as they enter the warehouse 2 time units later (see Figure 9(b)). In other words, for each edge-truck $\{i, j\}$, when both node-trucks i and j have been unloaded, there is no reason to wait for any other node-truck before loading $\{i, j\}$.

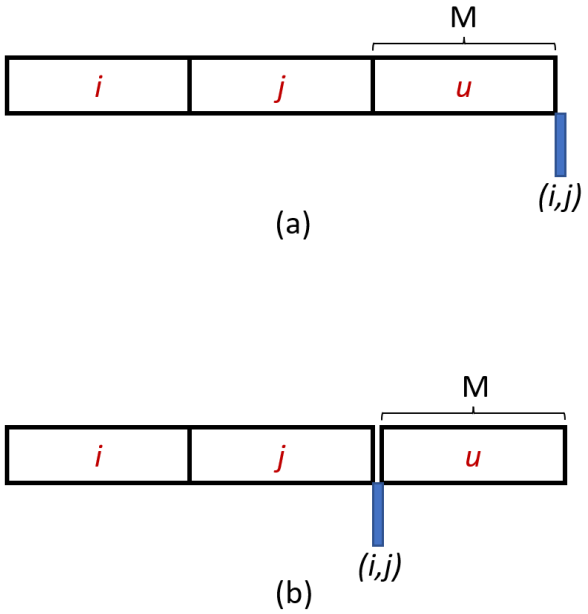

Figure 9: Structure of a feasible solution used for proof of Theorem 2.

In view of the above fact, given a truck schedule, consider the pallets which have to be transferred from node-trucks i and j to the edge-truck $\{i, j\}$, and suppose that $i \prec j$ in the schedule. First, considering j, the sojourn time of the pallet carried by j is decomposed into

a node-truck and an edge-truck contribution. The node-truck part is only M, i.e. the processing time of j, as this pallet is directly loaded in $\{i,j\}$ without any node-truck delaying the process. The edge-truck part is given by the number of edge-trucks scheduled between j and $\{i,j\}$ (each one contributes to the sojourn time with a value 1, as $\forall o \in O, p_o = 1$). Considering now i, the sojourn time of the pallet carried by i is also decomposed into a node-truck and an edge-truck contribution. Let $f(i)$ and $f(j)$ denote the positions of i and j among inbound trucks in the schedule (with $f(i) < f(j)$). The sojourn time of the pallets equals $M * (f(j) - f(i) + 1)$ (the node-truck part), plus the number of outbound trucks loaded between i and $\{i,j\}$ in the schedule (edge-truck part). Thus, the sojourn time of pallets loaded in $\{i,j\}$ is lower than:

$$M + m + M(f(j) - f(i) + 1) + m,$$

As there is a truck $\{i,j\}$ for each edge of the original graph, for a given truck schedule, the total sojourn time does not exceed:

$$\sum_{\{i,j\}\in E} (M + M(|f(j) - f(i)| + 1) + 2m) =$$
$$2mM + M \sum_{\{i,j\}\in E} |f(j) - f(i)| + 2m^2.$$

As a consequence, a truck schedule with a cost not exceeding $2mM + MK + 2m^2$ exists if and only if $f(\cdot)$ defines a linear arrangement of value not exceeding K. □

Figure 10 illustrates the reduction in Theorem 2. A graph with a node numbering is shown in Figure 10(a). Figure 10(b) shows the handover graph of the corresponding CTSP instance. Figure 10(c) represents a feasible schedule. For this instance, the flow times of the two pallets carried away by truck (b,c) are $3M + 3$ and M. For this example with $m = 10$, the linear arrangement displayed in Figure 10(a) has value $K = 16$ and the value of the overall flow time is $36M + 38$ ($M \geqslant 200$).

As in the above proof the processing times of inbound and outbound trucks do not respect Equation (4), the argument cannot be applied to establish the complexity of CTSP(1,Cor), which therefore remains open for general G_H.

3.2.2 *Two doors*

In this section, we focus on CTSP with $n = 2$. As the complexity increase much, we are focusing in particular on CTSP(2,Cor).

A key property of CTSP(2,Cor) is expressed in the following lemma.

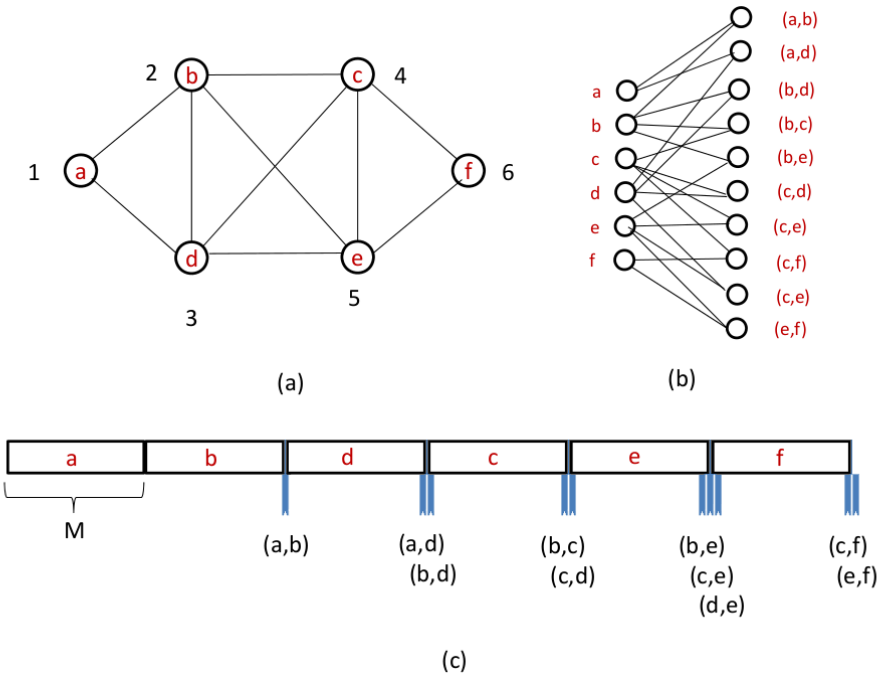

Figure 10: Illustration of the reduction of the proof of Theorem 2.

Lemma 3. *Given an instance of CTSP(2,Cor), let O_m be a set of outbound trucks assigned to a given door m, and suppose that they are scheduled consecutively, without idle time, from time r onwards. The contribution to the objective function of the trucks in O_m does not depend on their sequencing and is given by*

$$r \sum_{o \in O_m} p_o + \frac{1}{2}[(\sum_{o \in O_m} p_o)^2 - (\sum_{o \in O_m} p_o^2)].$$

Proof. Let us consider the objective function (5) having two terms, the first related to the outbound trucks, the second to the inbound trucks. Let focus on the first term: $\sum_{o \in O} p_o s_o$. If all trucks in O_m are scheduled consecutively in non-decreasing order of their index, the first one is scheduled at time r, ($s_0 = r$), the second one at $r + p_0$, the third one at $r + p_0 + p_1, \ldots$, hence:

$$\sum_{o \in O} p_o s_o = (r) * p_0 + (r + p_0)p1 + (r + p_0 + p_1)p2 + \ldots$$
$$+(r + p_0 + p_1 + \cdots + p_{|O_m|-1})p_{|O_m|}$$

which is equal to

$$\sum_{v \in O_m} (r + \sum_{u<v} p_u)p_v = \sum_{v \in S} rp_v + \sum_{u<v} p_v p_u$$

Since $(\sum_{v \in O_m} x_v)^2 = \sum_{v \in O_m} x_v^2 + 2\sum_{u<v} x_u x_v$, it follows that:

$$\sum_{o \in O} p_o s_o = r \sum_{o \in O_m} p_o + \frac{1}{2}[(\sum_{o \in O_m} p_o)^2 - (\sum_{o \in O_m} p_o^2)].$$

□

Consider now the following auxiliary problem, that we call (3/4,1/4)-PARTITION: Given a set O of n integers $p_1, p_2, \ldots, p_k$ and $W = \sum_{u \in O} p_u$, is there a partition (O_1, O_2) such that $\sum_{u \in O_1} p_u = \frac{3}{4}W$ and $\sum_{u \in O_2} p_u = \frac{1}{4}W$?

Lemma 4. *(3/4,1/4)-PARTITION is NP-complete.*

Proof. Obvioulsy, CTSP(2, Cor) is in NP. Lets consider an instance of the well-known PARTITION problem: Given k integers, $q_1, q_2, \ldots, q_k$, letting $H = \sum_{u \in O} q_u$, is there a partition (O_1, O_2) such that $\sum_{u \in O_1} q_u = \sum_{u \in O_2} q_u = \frac{H}{2}$?

To show that PARTITION reduced to (3/4,1/4)-PARTITION, we define an instance of (3/4,1/4)-PARTITION with $k+1$ integers $p_u, u = 1, \ldots, k+1$. The first k integers coincide with those of the instance of PARTITION, i.e., $p_u = q_u$, while $p_{k+1} = H$, so that $W = 2H$. Clearly, a solution to (3/4,1/4)-PARTITION exists if and only if a solution to PARTITION exists, as the former is simply obtained from the latter adding the integer p_{k+1} to one of the two sets.

□

We are now in the position of establishing the main result of this section.

Theorem 5. *CTSP(2,Cor) is NP-hard, even if G_H is a 1-biclique.*

Proof. We reduce (3/4,1/4)-PARTITION to CTSP(2,Cor). Given an instance of (3/4,1/4)-PARTITION, we define an instance of CTSP(2,Cor) as follows. There are 2 identical inbound trucks and k outbound trucks (so, $|I| = 2$ and $|O| = k$). The graph G_H is a 1-biclique, so $(i, o) \in A$ for all pairs (i, o). In view of (5), we specify directly the values p_i and p_o. Both the inbound trucks have a processing time $W/2$, while each outbound truck u has a processing time p_u, $u = 1, \ldots, k$. In what follows, let $P^{(2)}$ denote the sum of the squares of integers p_u, i.e., $P^{(2)} = \sum_{u \in O} p_u^2$. The problem is to determine whether there exists a schedule σ having an objective value not greater than the threshold value γ, where

$$\gamma = \frac{11}{16}W^2 - \frac{1}{2}P^{(2)}.$$

Concerning inbound trucks, either (i) the two inbound trucks are consecutively scheduled at the same door, or (ii) they are assigned to different doors. Call *asymmetric* and *symmetric* two schedules having structure (i) and (ii) respectively. Note that in both situations, due to the start-start precedence constraints, the first outbound truck(s) cannot start being loaded before time $W/2$. In what follows, we let O_1 and O_2 denote the subsets of outbound trucks scheduled at door 1 and 2 respectively.

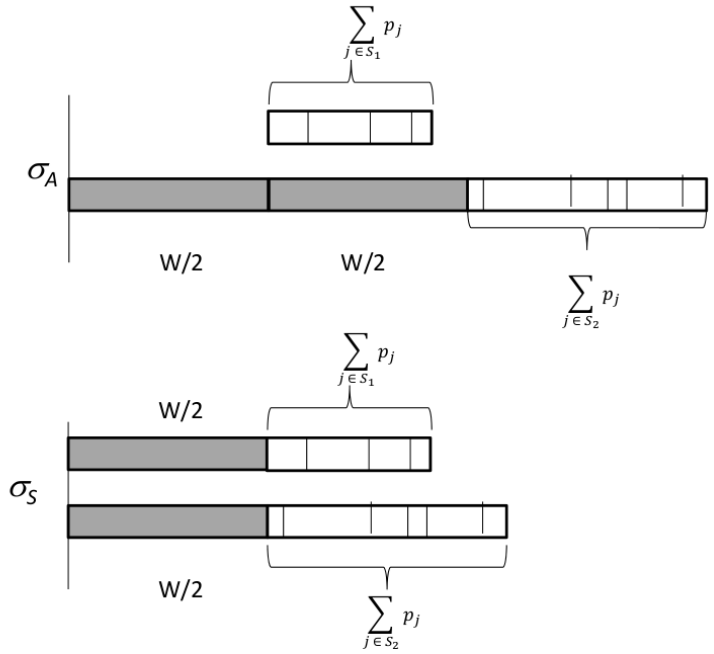

Figure 11: Asymmetric and symmetric schedules in the proof of Theorem 5.

Let us first analyze *asymmetric schedules*. With no loss of generality, we assume that the two inbound trucks are unloaded at door 2 (see Figure 11). Note that in an asymmetric schedule, the start times of two inbound trucks are 0 and $W/2$ respectively, so the contribution of inbound trucks to the objective function is $-\frac{W}{2}\frac{W}{2} = -W^2/4$. So, in an asymmetric schedule, outbound trucks start being scheduled at door 1 at time $W/2$, while the others start at door 2 at time W (see Figure 11). From Lemma 3, the contribution of outbound trucks in an asymmetric schedule is:

$$\begin{gathered}\tfrac{W}{2}\textstyle\sum_{o\in O_1} p_o + \tfrac{1}{2}(\sum_{o\in O_1} p_o)^2 - \tfrac{1}{2}\sum_{o\in O_1} p_o^2 + \\ W\textstyle\sum_{o\in O_2} p_o + \tfrac{1}{2}(\sum_{o\in O_2} p_o)^2 - \tfrac{1}{2}\sum_{o\in O_2} p_o^2 = \\ \tfrac{W}{2}\textstyle\sum_{o\in O_1} p_o + W\sum_{o\in O_2} p_o + \tfrac{1}{2}(\sum_{o\in O_1} p_o)^2 + \\ \tfrac{1}{2}(\textstyle\sum_{o\in O_2} p_o)^2 - \tfrac{1}{2}\sum_{o\in O} p_o^2.\end{gathered}$$

Therefore, the objective function value for an asymmetric schedule σ_A is

$$\begin{aligned}f(\sigma_A) = \frac{W}{2}\sum_{o\in O_1} p_o + W\sum_{o\in O_2} p_o + \frac{1}{2}(\sum_{o\in O_1} p_o)^2 \\ + \frac{1}{2}(\sum_{o\in O_2} p_o)^2 - \frac{1}{2}\sum_{o\in O} p_o^2 - \frac{W^2}{4}.\end{aligned}$$

Note that $f(\sigma_A)$ only depends on the way outbound trucks are partitioned between O_1 and O_2. The differentiation of $f(\sigma_A)$ shows that it

is minimal for $\sum_{o\in O_1} p_o = \frac{3}{4}W$ and $\sum_{o\in O_2} p_o = \frac{1}{4}W$, which gives the following lower bound:

$$LB_A = \frac{11}{16}W^2 - \frac{1}{2}P^{(2)}$$

Let us now consider *symmetric schedules*. In any such schedule σ_S, both inbound trucks start at time 0, so their contribution to the objective function is 0. Outbound trucks start at both doors at time $W/2$ (see Figure 11). Hence, the objective function $f(\sigma_S)$ only includes the contribution of outbound trucks which, from Lemma 3, is

$$f(\sigma_S) = \frac{W}{2}\sum_{o\in O} p_o + \frac{1}{2}(\sum_{o\in O_1} p_o)^2 + \frac{1}{2}(\sum_{o\in O_2} p_o)^2 - \frac{1}{2}\sum_{o\in O} p_o^2.$$

Again, simple calculus shows that a lower bound on $f(\sigma_S)$ is

$$LB_S = \frac{3}{4}W^2 - \frac{1}{2}P^{(2)},$$

which is attained if $\sum_{o\in O_1} = \sum_{o\in O_2} = \frac{W}{2}$.

Comparing the two lower bounds, we have

$$LB_S - LB_A = \frac{1}{16}W^2$$

which is strictly positive. As a consequence, since $LB_A = \gamma$, we have that a schedule of value γ exists if and only if there exists an asymmetric schedule such that $\sum_{o\in O_1} p_o = \frac{3}{4}W$ and $\sum_{o\in O_2} p_o = \frac{1}{4}W$. In turn, this schedule exists if and only if a corresponding solution to the instance of (3/4,1/4)-PARTITION exists. □

As CTSP(2,Cor) is a special case of CTSP(2,Unr) in which relations (2) and (3) hold, Theorem 5 implies immediatly the NP-hardness of CTSP(2,Unr).

3.2.3 *Synthesis of CTSP complexity*

Our complexity contributions are summarized in Table 1 where NPH stands for NP-Hard.

n	C , 1-biclique	C	U, 1-biclique	U
1	O(n)	open	O(n log n) (Th. 1)	NPH (Th.2)
2	NPH (Th.5)	NPH (Th.5)	NPH (Th.5)	NPH (Th.5)

Table 1: Complexity of CSTP problem

In view of the NP-hardness of CTSP in the general case for both the correlated and unrelated cases, the next section focuses on methodologies to solve problem instances.

4

INSTANCES OF THE PROBLEM

4.1 DEFINITION OF THE INSTANCES

In order to assess the potential of our various solving approaches, this section explains how we generated a set of instances. Due to our specific assumptions, the studied problem need its proper instances, which were not available in the literature

INSTANCE PARAMETERS Our instances are defined by various parameters:

- the number of doors n
- the number of trucks $|U| = |I| + |O|$
- the number of inbound trucks $|I|$
- the number of outbound trucks $|O|$
- the processing times of all trucks p_u, $u \in U$
- the release dates of all trucks r_u, $u \in U$
- the deadlines of all trucks d_u, $u \in U$
- the loads of all trucks w_u, $u \in U$
- the handover relations, i.e. the precedences between inbound-outbound trucks P

This data represents information known by the crossdock manager, and allows to compute a solution. An instance commonly corresponds to a single working day. A correlated instance is illustrated previously in Figure 5.

The selection of the data allows various kinds of instances: to begin, the scale of the problem is 24 hours. A schedule is realized for every day and we are considering that all pallets are unloaded and loaded in the same day (no pallets left for the next day). We also choose to work with a precision of 5 minutes assuming this is the time needed to load or unload one pallet. For this reason, all values are multiples of 5 min units. The horizon is 24h, so $T = 288$ as $288 \times 5 = 1440$ min.

The number of doors is chosen in the set $n \in \{1, 2, 5, 10\}$. We do not consider larger instances because of the complexity of the problem: we want to be able to solve the instances in reasonable time.

For a specific number of doors, several number of trucks are specified. The number of trucks is $|U| \in \{6n, 8n, 10n, 12n, 14n\}$. The number of inbound trucks is equal to the number of outbound trucks $|I| = |O|$. We fix the workload of the crossdock to 65%. This workload corresponds to the average occupation rate of the doors. To reach this workload, the average length of loading/unloading is calculated based on the number of trucks. Thus the loads (and the processing times) of the trucks are randomly generated within a predefined interval. Commonly, a link exists between load and processing time: the more pallets need to be (un)loaded, the more time is needed. However, depending on the truck, the size of the pallets, the category of products, this link between load and processing time can be more or less strict. For this reason, we will consider two kinds of instances: correlated and unrelated ones. In correlated instances, the truck processing time is proportional to the number of pallets. In order to respect the average targeted workload of the crossdock, for the correlated instances, the load of the $\{6n, 8n, 10n, 12n, 14n\}$ trucks is randomly picked in $\{[30, 30]; [17, 30]; [7, 30]; [2, 30]; [2, 25]\}$, respectively. The unrelated instances are obtained by modifying the processing times of the correlated instances. This operation is partially randomized: new values are generated around the initial ones, while conserving the average workload previously fixed. Concerning the handover relations, we assume that pallets of inbound truck i need to be loaded in $\{1, \ldots, \frac{p_i}{5}\}$ outbound trucks.

The generation of the time windows is done in two steps: the generation of tight time windows, called (0.00), and the generation of four larger time windows (0.25, 0.50, 0.75 and 1.00). These time windows are obtained form the (0;00) ones by proportionally decreasing the release date and increasing the latest departure date. We define a change by 0% as keeping the original tight time windows and a change by 100% as enlarging to the largest time windows. In fact, for the latter case, there are no time windows anymore as all release dates are set equal to the beginning of the planning horizon and all deadlines are set equal to the end of the planning horizon. The tight time windows are obtained as follows. The ready times r_i of inbound trucks are uniformly distributed in $[0, 0.9\frac{\sum p_u}{n}]$. The deadlines $\tilde{d}_o$ of outbound trucks are uniformly distributed in $[\Omega_o, H]$, with $\Omega_o = max_{(i,o)\in A}\{r_i + p_o\}$. The ready times r_o of outbound trucks are uniformly distributed in $[r_i^{max}, \tilde{d}_o - p_o]$, with $r_i^{max} = max_{(i,o)\in A}\{r_i\}$. The deadlines $\tilde{d}_i$ of inbound trucks are uniformly distributed in $[1.5(r_i + p_i), \tilde{d}_i^{max}]$, with $\tilde{d}_i^{max} = min\{min_{(i,o)\in A}\{\tilde{d}_o - p_o\} + p_i, max_{(i,o)\in A}\{\tilde{d}_o\}\}$.

For each combination of these parameters, 10 instances are generated. A set of 1000 correlated/unrelated instances is obtained. 550 additional instances with one door were added to further examine the complexity of the single door case.

4.2 TIME LIMITATION

In order to evaluate the difficulty to solve these instances, and being able to get relevant results within an appropriate time limit, we are now interested in the time needed to converge. We used the correlated version of the instances, with the time-indexed formulation, developed from (Berghman et al., 2015), and also detailed in the next chapter, as in general the time-indexed formulation is known for being time consuming. After 2 hours of computing (for a 24h schedule), some solutions still improve, but the improvement tends to be very negligible with respect to the time invested. Among the 1000 computed instances, a majority converged. To monitor the evolution of the solution over time, we interrupted calculation at 8 different time thresholds: 1, 2, 5, 10, 15, 30, 60 and 120 minutes. For each threshold, the following table presents the improvement of the UBs, relatively to those obtained at the previous stop.

The results are divided according to the number of doors and the tightness of the time windows, which impact the resolution of the instances. Focusing on the number of doors, the evolution of the solution values (upper bounds) from a selected set of difficult instances is detailed in Table 2.

Table 2: Evolution of upper bound for different number of doors

	Time (min)						
Doors	2	5	10	15	30	60	120
1	1%	0%	2%	0%	0%	0%	0%
2	5%	0%	9%	1%	1%	0%	1%
5	6%	0%	8%	3%	4%	4%	2%
10	8%	0%	13%	3%	9%	16%	11%

Based on the number of doors, the convergence of 1, 2 and 5 doors instances can be considered as stable after 15 min, since the following improvements of the solution are really low. The convergence of the 10 doors instances is partial, as the variation of the objective is still relevant (11% between 60 and 120 min).

Focusing on the tightness of the time windows, the evolution of the upper bound is detailed in Table 3. We notice slower convergence on larger time windows.

The tightest case reveals an unexpected situation as one minute is sufficient to find a solution which cannot be improved with 120× more time. Cases with large time windows, i.e. (0.75) and (1.00), require 2h of computation.

Considering both the number of doors and the tightness of the time windows highlights the convergence of all instances, excepting three categories, as depicted in Table 4.

Table 3: Evolution of upper bound for different time windows tightness

TW	Time (min)						
	2	5	10	15	30	60	120
0	0%	0%	0%	0%	0%	0%	0%
0,25	6%	0%	10%	0%	1%	1%	0%
0,5	7%	0%	11%	2%	6%	5%	4%
0,75	7%	0%	9%	4%	5%	8%	6%
1	5%	0%	9%	3%	5%	11%	7%

Table 4: Evolution of upper bound for different number of doors and time windows tightness

		Time (min)						
Doors	TW	2	5	10	15	30	60	120
1	0	0%	0%	0%	0%	0%	0%	0%
1	0,25	0%	0%	0%	0%	0%	0%	0%
1	0,5	2%	0%	2%	0%	0%	0%	0%
1	0,75	2%	0%	4%	0%	0%	0%	0%
1	1	3%	0%	2%	0%	0%	0%	0%
2	0	0%	0%	0%	0%	0%	0%	0%
2	0,25	2%	0%	2%	0%	0%	0%	0%
2	0,5	9%	0%	12%	2%	3%	1%	1%
2	0,75	9%	0%	14%	2%	1%	1%	1%
2	1	6%	0%	15%	2%	1%	0%	1%
5	0	0%	0%	0%	0%	0%	0%	0%
5	0,25	8%	0%	4%	1%	2%	0%	0%
5	0,5	8%	0%	13%	2%	4%	6%	5%
5	0,75	6%	0%	9%	5%	6%	8%	3%
5	1	6%	0%	12%	7%	8%	6%	3%
10	0	0%	0%	0%	0%	0%	0%	0%
10	0,25	15%	0%	32%	0%	3%	2%	1%
10	0,5	10%	0%	15%	2%	19%	14%	11%
10	0,75	10%	0%	9%	10%	13%	25%	21%
10	1	5%	0%	7%	4%	11%	39%	23%

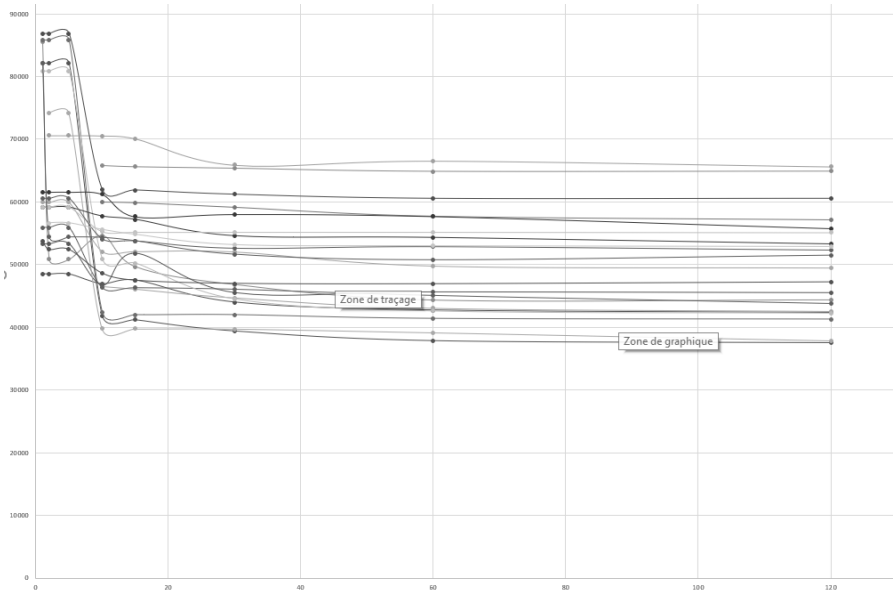

Figure 12: Upper bound evolution of a selection of the 10 doors, 0.25 time windows instances

The three concerned categories are 10 doors with (0.5), (0.75) and (1.00) TW. The improvement of the UB is still relevant, even after two hours of computing, with a respective improvement of 11%, 21% and 23%.

Only a few categories are not converging after a time of computation of 2 hours. Moreover, further increasing the computation time is not ideal as new improvement might arrive late.

Figure 12 illustrates the evolution of a representative part of the instances, with the biggest number of doors (10) and slightly relaxed time windows (0.25). The convergence can be easily certified. Remark that the curves representing the objective values of the solutions found over time are all decreasing overall, however we can observe that some curves are locally increasing. This phenomenon is due to the set-up of the experiment, and the non-deterministic solutions found by cplex within a predefined time limit. Intuitively, one would expect to interrupt cplex after a predefined time limit, report the solution, and relaunch for an additional time. However, due to technical issues, cplex does not relaunch the solving exactly where it is stopped leading to inexact time management. For that reason, each solution on the graph is obtained by a separate cplex run with a pre-defined time limit. In some cases (especially when the time limit is low), according to branch and hardware conditions, the solution found can be lower or higher than expected.

Part II

SOLVING THE CTSP

This second part introduces the proposed resolution methods to solve the CTSP problem. First, a general time indexed formulation is introduced, that can be implemented with a commercial solver like CPLEX. It will serve as a reference to assess the other methods. Secondly, a quadratic formulation, that can solve instances with a single door is proposed, even though the relevance of this formulation is quite reduced. Next, a critical set formulation, enforced by valid inequalities and a constraint programming formulation are developed. These formulations are designed to solve real-scale instances with dozens of doors.

5

TIME INDEXED FORMULATION

In this chapter we introduce a time indexed formulation for CTSP.

5.1 THE CHOICE OF TIME INDEXED MILP TO SOLVE CSTP

As introduced previously, the time indexed formulation introduced in Berghman et al. (2015) will be further explored in this chapter, in order to compare to new methods. This will allow us to have an idea about the difficulty of the instances, and measure the relative efficiency of more complex formulations. It is well known that time indexed formulations perform well for scheduling problems because the linear programming relaxations typically obtain strong lower bounds (Dyer and Wolsey, 1990) provided that the problem size remains reasonable.

5.2 FORMULATION OF THE TIME INDEXED MODEL

In the following, we will refer to the presented time indexed formulation as "TI". The time horizon is discretized into elementary intervals of unit length. We define $\mathcal{T}$ as the set of all time intervals, and τ as the generic time interval $[t-1, t[$ where t represents a moment in time.

For all trucks $u \in U$ and for all time periods $\tau \in \mathcal{T}$, we have

$$x_{u\tau} = \begin{cases} 1 & \text{if the unloading/loading of truck } u \text{ starts} \\ & \text{during time period } \tau, \\ 0 & \text{otherwise.} \end{cases}$$

For all trucks $u \in U$, $x_{u\tau} = 0$ if $\tau \leqslant r_u$ or if $\tau > \tilde{d}_u - p_u + 1$ as during these time intervals it is not possible to start unloading/loading truck u.

A time-indexed formulation for the considered CTSP problem, that we call TI_1 is the following:

$$\min \quad z = \sum_{(i,o)\in A} \sum_{\tau\in\mathcal{T}} w_{io}\tau\,(x_{o\tau} - x_{i\tau}) \tag{10}$$

subject to

$$\sum_{\tau \in \mathcal{T}} x_{u\tau} = 1 \qquad \forall u \in U \tag{11}$$

$$\sum_{\tau \in \mathcal{T}} \tau \, (x_{i\tau} - x_{o\tau}) \leqslant 0 \qquad \forall (i,o) \in A \tag{12}$$

$$\sum_{u \in U} \sum_{b=\tau-p_u+1}^{\tau} x_{ub} \leqslant n \qquad \forall \tau \in \mathcal{T} \tag{13}$$

$$x_{u\tau} \in \{0,1\} \qquad \forall u \in U; \forall \tau \in \mathcal{T} \tag{14}$$

The objective function (10) minimizes the total sojourn time. Constraints (11) demand that each truck is assigned to exactly one door. Constraints (12) ensure that if inbound truck i and outbound truck o are connected, the loading start time for truck o is not earlier than the unloading start time for truck i. Constraints (13) enforce the capacity of the doors.

Alternatively precedence constraint (12) can be expressed as follow:

$$\sum_{b=1}^{\tau} (x_{ib} - x_{ob}) \geqslant 0 \qquad \forall (i,o) \in A; \forall \tau \in \mathcal{T} \tag{15}$$

These constraints state that a loading task can only begin after all preceding unloading tasks have been started. These constraints are called *disaggregated* (Christofides et al., 1987). The formulation obtained by replacing constraints (12) in formulation TI_1 by (15) will be referred to as formulation TI_2. TI_2 is stronger than TI_1, since each feasible solution for the LP relaxation of TI_2 is also a feasible solution to the LP relaxation of TI_1. However, TI_2 contains $|A||\mathcal{T}|$ constraints (15), while TI_1 includes only $|A|$ constraints (12). The additional CPU time needed to solve the larger linear program can significantly counterbalance the improvement of the bound. Computational experiments have been conducted to compare the efficiency of both formulations. We will detail them in the following section.

5.3 COMPARISON OF TI FORMULATIONS

Both formulations have been tested on two predefined sets of 1000 instances, correlated and unrelated versions.

To measure their performances, we focus on the elements in the table below:

- Optimal (*Opt.*): the number of instances for which an optimal solution is found by the algorithm before reaching the time limit.
- Feasible (*Feas.*): the number of instances for which a feasible solution is found. It may be the optimal solution, but the algorithm cannot prove the optimality within the allowed computation time.

- Unfeasible (*Unfeas.*): the number of instances that are proved infeasible.
- Unknown (*Unkn.*): the number of instances for which neither a feasible solution nor a proof of unfeasibility is found within the time limit.
- Computation time (*Time*): the average time in seconds needed to prove optimality or infeasability whenever both TI_1 and TI_2 succeeded in proving.
- Gap (*Gap*): the average gap after reaching the time limit whenever both TI_1 and TI_2 succeeded to find a solution.
- Relative upper bound (*relative UB*): the average difference between two upper bounds as a percentage of the best upper bound, whenever a feasible or optimal solution is found before the end of the computation time. Considering UB_a and UB_b: relative $UB_a = \frac{UB_a - UB_b}{\max\{UB_a, UB_b\}}$. As such, the formulation with best UB has a positive *relative UB*.

The number of optimal and infeasible instances is obviously important as the more a method is able to find, the more efficient. Also, the number of unknown instances is relevant because an efficient method should establish the possibility of solving the instance in short time. The number of feasible instances is more tricky to interpret, as finding feasible (but not optimal) solutions on *easy* instances is negative, while finding feasible solutions on *hard* instances is positive. Anyway, the number of feasible instances is a third indicator that simply completes the number of optimal, infeasible and unknown instances.

Due to similar behaviour, the table establishing the comparison between formulations cumulates both correlated and unrelated results.

NB: Instances for which optimality or infeasability was not proven by at least one of both formulations do not participate to the Time calculation. Instances for which no feasible solution was found by at least one of both formulation does not participate to the Gap *and* relative UB *calculation.*

The results presented in Table 5 show that the obtained solutions are close from one formulation to another, with the same number of unfeasible proved and few differences on the number of *Optimal* found (1.5%), *Feasible* found (0.3%) and *Unknown* (3 instances versus 1). On the one hand TI_2 seems to be faster that TI_1 to resolve instances, on the other hand the value *relative UB* is -20% on average, and the difference increases with the number of doors. Because of the similar gap of the formulations, a better upper bound for TI_1 means a worse lower bound. However, we grant a strong importance to the solution value: the UB, and the improvement of the UB. For this reason the formulation TI_1 is selected as the final version of the time indexed formulation and will be used in the following of this work.

n	form.	*Opt.*	*Feas.*	*Unfeas.*	*Unkn.*	*Time (s)*	*Gap*	*relative UB*
all	F1	392	1362	243	3	454	60.9%	20.0%
all	F2	398	1358	243	1	372	59.6%	-20.0%
1	F1	190	216	94	0	579	37.4%	0.4%
1	F2	184	222	94	0	516	34.3%	-0.4%
2	F1	87	343	70	0	461	67.5%	12.1%
2	F2	95	335	70	0	394	67.3%	-12.1%
5	F1	49	400	51	0	133	69.5%	30.5%
5	F2	51	397	51	1	110	68.2%	-30.5%
10	F1	66	403	28	3	432	66.8%	34.0%
10	F2	68	404	28	0	206	66.1%	-34.0%

Table 5: Comparison of formulations TI_1 and TI_2 with time limit of 2h

Although it is not presented in the table, without a surprise, for the same instances, returned status (optimal/feasible/unfeasible) are mainly identical for both formulations. On the maximum of 398 *Optimal* found, 374 are the same instances (94%). All *unfeasible* are found by both of the formulations. We can note that the instance *Unknown* for TI_2 is different from the 3 instances *Unknown* for TI_1, but this is a very small number.

5.4 PERFORMANCES AND LIMITS OF THE TIME INDEXED FORMULATION

To analyse the performance of the TI formulation, we detail the results with respect to both the number of doors (n) and the average number of trucks per door ($\frac{k}{n}$) in Tables 6 and 7. The gap is based on the best UB and the best LB found by CPLEX in the predefined computation time, and is calculated as follows: $Gap = \frac{UB-LB}{UB}$, for all instances with a feasible solution.

The first thing we can highlight is the efficiency of the TI formulation to find feasible solutions. Very few of the instances remain unknown. On the smallest instances, with one door and six trucks, the method performs very well, but with slightly more trucks, the number of *Feasible* increases drastically at the expense of *Optimal* ones, and the average gap increases to ~30%. When the number of doors or the number of trucks per door increases, the number of optimal solutions continues to decrease quickly while the average gap reaches ~70%.

Unexpectedly, the number of optimal solutions and the average gap seem to be stable when the instances reach a certain size. The number of optimal and feasible solutions are better for one door, and for two doors with 6 or 8 trucks per door. Anywhere else, the number of optimal and feasible solutions does not differ much. The Gap for instances with one door and a low number of trucks per door are

n	$\frac{k}{n}$	*Opt.*	*Feas.*	*Unfeas.*	*Unkn.*	*Time (s)*	*Gap*
all	all	189	685	124	2	5110	61,5%
1	6	38	2	10	0	1717	0,7%
1	8	19	22	9	0	3530	30,8%
1	10	12	29	9	0	4408	49,6%
1	12	13	28	9	0	4153	55,8%
1	14	11	31	8	0	4552	60,7%
2	6	11	31	8	0	4480	62,0%
2	8	12	31	7	0	4676	66,0%
2	10	7	35	8	0	5206	69,7%
2	12	7	38	5	0	5553	68,1%
2	14	5	38	7	0	5538	71,5%
5	6	4	39	7	0	5685	71,3%
5	8	2	40	8	0	5765	73,9%
5	10	4	40	6	0	5797	71,4%
5	12	6	40	4	0	5765	67,8%
5	14	8	40	2	0	5826	65,3%
10	6	8	40	2	0	5765	65,4%
10	8	6	40	4	0	5906	68,8%
10	10	5	40	5	0	5786	69,4%
10	12	6	40	3	1	6019	66,0%
10	14	5	41	3	1	6069	66,4%

Table 6: TI results on correlated instances

acceptable. But for all other instances the gaps are around 60~70%, independently of the number of doors or trucks per door. We can notice that the time required to solve the instances is increasing overall. As expected, to perform similarly, TI need more time to solve instances with an increasing number of doors and trucks. The efficiency to find solution even for the worst considered case confirms the relevance of the time limit.

Although the TI formulation seems efficient on the proposed instances to find solutions (low number of *Unknown*), the gaps of numerous *feasible* solutions are really important, and show the limit of the TI formulation to find good solutions (UB is too high), or to prove the quality of the solution (LB is too low). These conclusions seem insensitive to the nature (correlated versus unrelated) of the problem.

Theses limits are a strong motivation to explore new methods.

n	$\frac{k}{n}$	*Opt.*	*Feas.*	*Unfeas.*	*Unkn.*	*Time (s)*	*Gap*
all	all	203	677	119	1	5075	60,2%
1	6	37	2	11	0	1146	0,6%
1	8	21	17	12	0	3184	18,1%
1	10	15	25	10	0	3967	41,0%
1	12	12	30	8	0	4460	49,9%
1	14	12	30	8	0	4385	61,6%
2	6	10	30	10	0	4473	64,4%
2	8	10	32	8	0	4728	67,7%
2	10	9	34	7	0	5224	68,9%
2	12	9	37	4	0	5486	66,8%
2	14	7	37	6	0	5552	69,2%
5	6	4	40	6	0	5766	72,0%
5	8	5	40	5	0	5775	69,1%
5	10	3	40	7	0	5768	73,0%
5	12	6	40	4	0	5783	67,5%
5	14	7	41	2	0	5978	65,5%
10	6	9	40	1	0	5768	65,2%
10	8	6	41	3	0	6038	68,1%
10	10	6	40	4	0	5829	68,8%
10	12	9	40	1	0	5898	64,5%
10	14	6	41	2	1	6288	65,8%

Table 7: TI results on unrelated instances

6 QUADRATIC FORMULATION

In this chapter, we introduce a quadratic formulation for the CSTP with only one door and no binding time windows. The 0/1 quadratic formulation is widely studied and has a relatively simple structure. So although it can only be used for a very special case, we think it is worth presenting the formulation and compare the computational results with the other formulations.

6.1 A QUADRATIC FORMULATION FOR THE SINGLE-DOOR PROBLEM WITHOUT TIME WINDOWS

This special case can be modeled using a straightforward quadratic formulation, which is a variation of the linear formulation of (Potts, 1980).

For all trucks $(u, v) \in U$, the following decision variables are set up:

$$x_{uv} = \begin{cases} 1 & \text{if truck } u \text{ precedes truck } v \text{ in the schedule,} \\ 0 & \text{otherwise,} \end{cases}$$

Remark that x_{uv} is also 1 if there are some other trucks scheduled between u and v, so not only in the case where they are scheduled immediately the one after the other.

As we only have one door, and as the sojourn time of a pallet is calculated as the difference between the starting time of the corresponding unloading activity and the starting time of the corresponding loading activity, the sojourn time of a pallet is at least equal to p_i. Moreover, for each truck u (either inbound or outbound) sequenced *between* an inbound truck i and an outbound truck o, the w_{io} pallets will spend an extra p_u time in the warehouse. Hence, the total time the pallets spend in the warehouse after they are unloaded and before they are loaded can be formulated as the processing time of all trucks u such that $x_{iu} = 1$ and $x_{uo} = 1$, and this processing time contributes to the sojourn time of the w_{io} pallets unloaded from i and loaded on o. This does not hold anymore when we have release dates as those might add some idle time in between the (un)loading of two trucks. This formulation cannot be used if more than one door is considered, or with time windows: the proposed objective formulation is only correct when there is no idle time. With one door, without time widows, the optimal solution does not present idle time, and thus can be represented by a continuous sequence of trucks. If more doors

are considered, idle times can be needed due to precedence relations between connected trucks. And if release dates are considered, idle times can be needed to respect the time windows.

A quadratic formulation for the considered special case of the CTSP problem is the following:

$$\min \quad z = \sum_{i \in I} w_i p_i + \sum_{i \in I} \sum_{o \in O} \sum_{u \neq i,o} w_{io} p_u x_{iu} x_{uo} \tag{16}$$

subject to

$$x_{uv} + x_{vu} = 1 \qquad \forall u, v \in U \tag{17}$$

$$x_{uv} + x_{vj} + x_{ju} \leqslant 2 \qquad \forall u, v, j \in U \tag{18}$$

$$x_{io} = 1 \qquad \forall (i, o) \in A \tag{19}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u, v \in U \tag{20}$$

The total sojourn time is minimized in (16). The first term corresponds to the sojourn time of each pallet during their unloading, with $w_i = \sum_{o \in O} w_{io}$; the second term adds the sojourn time of the pallets after the truck being unloaded, and before the start of the loading of the pallets. Constraints (17) impose an order between any pair of 2 trucks and (18) prevent cycles. If u precedes v ($x_{uv} = 1$) and v precedes j ($x_{vj} = 1$), then j precedes u ($x_{ju} = 1$ and hence $x_{uj} = 0$). Finally, constraints (19) ensure that outbound trucks do not precede their connected inbound trucks.

6.2 PERFORMANCES AND LIMITS OF QUADRATIC FORMULATION

Due to the necessary characteristics of the problem to be able to use the quadratic formulation, only 50 instances from the generated set in chapter 4 are eligible. From the original set, we select the instances with one door, and without time windows. We add to this set 110 new instances, by increasing the number of trucks, to better explore the behaviour of this formulation. The time limit is fixed to 2h, but the actual computation time needed by the solver is shorter.

The instances are computed using the quadratic solver of CPLEX 12.8. The results are partitioned according to the number of trucks per door in Tables 8 and 9. All the instances are optimally solved, so *Feasible*, *Unfeasible* and *Unknown* are not mentioned in the table. *Opt.*, *Time* and *Gap* are calculated as explained in the previous chapter.

As the results of the correlated and the unrelated instances are highly similar, following analysis applies for both sets.

The quadratic formulation works very well. All the instances are solved to optimality. When the number of trucks per door is small (below 18), the optimum is found very quickly, in less than 0.1s, for both correlated and uncorrelated instances. A few seconds are needed

n	k	*Opt.*	*Time (s)*	*gap*
1	6	10	<0.1	0%
1	8	10	<0.1	0%
1	10	10	<0.1	0%
1	12	10	<0.1	0%
1	14	10	<0.1	0%
1	16	10	<0.1	0%
1	18	10	0,1	0%
1	20	10	0,7	0%
1	22	10	0,8	0%
1	24	10	2,1	0%
1	28	10	6,0	0%
1	32	10	14	0%
1	36	10	28	0%
1	40	10	43	0%
1	44	10	188	0%
1	48	10	358	0%

Table 8: Quadratic formulation on correlated instances

n	k	*Opt.*	*Time (s)*	*gap*
1	6	10	<0.1	0%
1	8	10	<0.1	0%
1	10	10	<0.1	0%
1	12	10	<0.1	0%
1	14	10	<0.1	0%
1	16	10	<0.1	0%
1	18	10	0,1	0%
1	20	10	0,7	0%
1	22	10	0,8	0%
1	24	10	2,1	0%
1	28	10	6,0	0%
1	32	10	14	0%
1	36	10	29	0%
1	40	10	43	0%
1	44	10	188	0%
1	48	10	358	0%

Table 9: Quadratic formulation on unrelated instances

with more trucks per door (until 40) and when we consider more than 40 trucks, the computation time increases considerably but stays low compared to the time limit.

The results of the quadratic formulation have to be compared to the results of the time indexed formulation. Chapter 9 is dedicated

to this comparison, with additional results from other formulations, proposed in the following chapters.

To conclude, we can say that the quadratic formulation is as expected very efficient for instances with one door and without time windows. However these required conditions drastically limit its use. A method with similar results able solve the general problem would be desired.

7

CRITICAL SETS FORMULATION

The CTSP is complex mainly due to the highly cumulative nature of the resource constraints (i.e., the doors of the cross-dock). Therefore, designing efficient methods for solving industrial large-size CTSP is suitable. In this section, we introduce a new approach that relies on the basic critical set concept. We show how valid inequalities can be progressively exhibited to enforce a relaxed MIP formulation of the CTSP. We also explore different strategies to analyze the efficiency of this branch-and-cut method.

7.1 BASIC CRITICAL-SETS FORMULATION

CRITICAL SET The concept of a critical set is well-known in the scheduling literature (Demeulemeester and Herroelen, 2002). A set of scheduled tasks/trucks is said critical if, due to mutually overlapping intervals, its implementation requires more resources than available. The critical set e has two proprieties. The tasks/trucks belonging to e use a common set of doors (need a certain working capacity) and their simultaneous processing induces a workload higher than the offered capacity. A solution with such critical set can become feasible, if trucks from the critical set are arranged so that they will not consume too much resource at the same time. On the opposite, a solution violates the critical set if the resource consumption of these trucks exceeds the resource capacity. Figure 13 illustrates this concept considering a CTSP with two doors and four trucks to unload/load. In solution (i), one can point out the critical set of size 4 ($\{a, b, c, d\}$ or alternatively consider four critical sets of size 3 ($\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$ and $\{b, c, d\}$). Case (ii) illustrates only two critical sets of size 3 ($\{a, c, d\}$ and $\{b, c, d\}$), (iii) only one critical set ($\{b, c, d\}$), and (iv) has no critical set. A critical set is said *minimal* if, after removing a single element from the set, it is not critical anymore. For instance, in figure 13, solution (i) contains four minimal critical sets ($\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$ and $\{b, c, d\}$), solution (ii) has two minimal critical sets ($\{a, c, d\}$ and $\{b, c, d\}$), and solution (iii) presents only one minimal critical set ($\{b, c, d\}$). Each truck requiring exactly one door (during the time p_u defined for each truck $u \in U$), a set of n trucks or less cannot be obviously critical for n doors, as it indubitably meets the resource requirement. Thus, a critical set has a cardinality strictly greater than n. A minimal critical set has a cardinality of $n + 1$.

Before introducing valid inequalities, let us first present the original critical-sets formulation of the problem. To model end-to-start

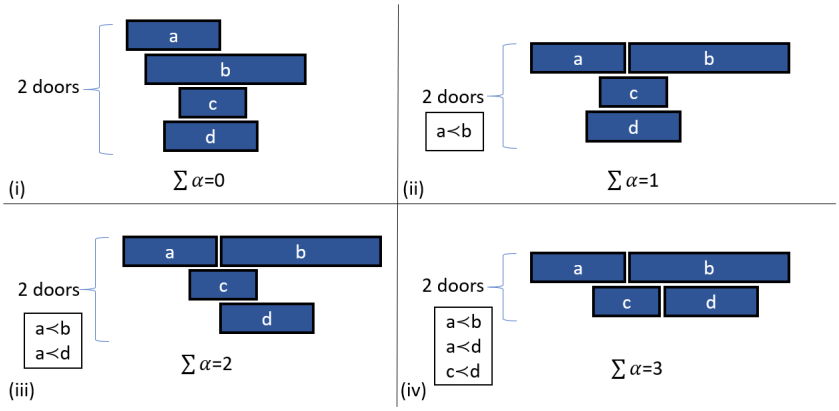

Figure 13: Possible solutions from a critical set of 4 trucks, according to imposed precedences

constraint precedences between trucks, this formulation uses binary variables α_{uv} for all $(u, v) \in U^2$ with:

$$\alpha_{uv} = \begin{cases} 1 & \text{if } u \text{ ends before the starting of } v \ (u \prec v) \\ 0 & \text{otherwise} \end{cases}$$

In other words, $\alpha_{uv} = 1$ if $s_u + p_u \leqslant s_v$. This kind of precedence constraint should obviously be set when u and v are allocated to the same door. It can also be indirectly satisfied when u and v are allocated to different doors, for instance u is allocated on a door, while v starts to be handled after the end of u on another door. For any pair of non-overlapping trucks (u, v), a precedence between u and v exists: $\alpha_{uv} + \alpha_{vu} = 1$. In the case of overlapping trucks (u, v), no end-to-start precedence exist: $\alpha_{uv} + \alpha_{vu} = 0$.

Using a critical sets formulation, the CTSP objective is still written as follows:

$$\min \sum_{i \in I} \sum_{o \in O} w_{io}(s_o - s_i) \tag{21}$$

In the correlated case CSTP(C, n), due to the proportional relation between w_u and p_u, the objective can be simplified as:

$$\min \sum_{o \in O} s_o p_o - \sum_{i \in I} s_i p_i \tag{22}$$

The CSTP is subject to:

$$s_o - s_i \geqslant 0 \qquad \forall (i,o) \in A \quad (23)$$

$$s_u \geqslant r_u \qquad \forall u \in U \quad (24)$$

$$-s_u \geqslant p_u - \tilde{d_u} \qquad \forall u \in U \quad (25)$$

$$s_v - s_u + \alpha_{uv}(M_{uv} - p_u) \geqslant M_{uv} \qquad \forall (u,v) \in U^2, u \neq v \quad (26)$$

$$\sum_{(u,v)\in e^2, u\neq v} \alpha_{uv} \geqslant |e| - n \qquad \forall e \in E \quad (27)$$

$$\alpha_{uv} \in \{0,1\} \qquad \forall (u,v) \in U^2, u \neq v \quad (28)$$

$$s_u \in \mathcal{R} \qquad \forall u \in U \quad (29)$$

The big M constraints (26) are defined by $M_{uv} = p_u - d_u + r_v$. E is the set of all critical sets.

Constraints (23) model start-start precedences due to handover relationships between inbound and outbound trucks sharing a flow of pallets. Constraints (24) and (25) impose the respect of release dates and deadlines, respectively. Constraints (26) and (27) model the resource constraints using the concept of critical sets. Constraints (26) link the start variables of the trucks with precedence variables α using big-M constraints. M_{uv} is defined such that $s_v - s_u \geqslant M_{uv}$. Such constraints do not affect the model when $\alpha_{uv} = 0$. However, $\alpha_{uv} = 1$ forces end-to-start precedences between u and v as we obtain $s_v - s_u \geqslant p_u$. Constraints (27) express that, for each critical set e of cardinality $|e|$, a minimum number of $|e| - n$ precedence constraints should be imposed between pairs of trucks belonging to e, which ensures that the set e of trucks will never induce a resource consumption greater than n doors. Constraints (27) intentionally consider *all* critical sets, even though it is well known that one can restrict only to minimal ones, i.e.:

$$\sum_{(u,v)\in e^2, u\neq v} \alpha_{uv} \geqslant 1 \qquad \forall e \in E_{min} \quad (30)$$

with $E_{min} \subset E$ the subset of all minimal critical sets.

The number of critical sets, minimal or not, is exponential, which makes the formulation itself exponential in size and very ineffective in practice. That is why we suggest to relax all constraints (27), and to progressively re-introduce them, using specific cuts, as explained in the next section.

7.2 NEW VALID INEQUALITIES

The following part introduces a valid inequality, which even strengthens constraints (27).

DEFINITION In a MIP, a *valid* inequality is a constraint that does not cut off any integer solutions. If this inequality is able to con-

sider less continuous solutions than the original formulation, while conserving all integer solutions, it is called a *strong* valid inequality (Wolsey, 1998). A strong valid inequality enforces the original formulation, leading to an easier resolution while reducing the search space. The inequality, to effectively enforce the formulation, cannot be a linear combination of existing constraints. Figure 14 presents a set of solutions. Integer solutions are represented by explicit points. Constraints c_1 and c_2 are valid inequalities, but only c_1 enforces the model, wiping out part of the continuous solutions. As constraint c_3 is wiping out at least one integer solution, materialized with point p_1, it is not valid.

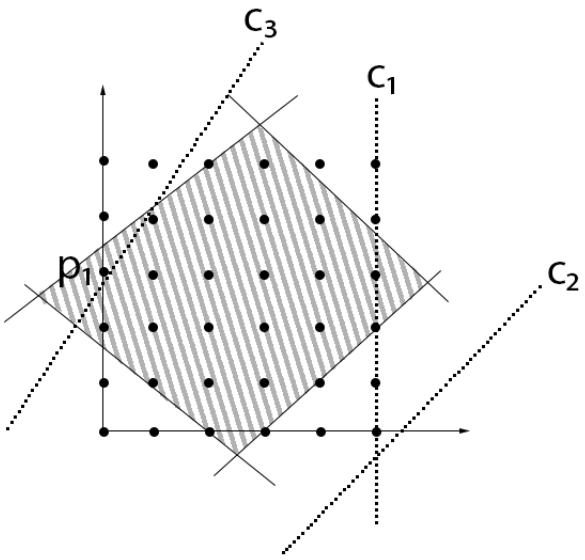

Figure 14: Illustration of valid inequality

In the sequel of this section, we aim to replace the exponential number of resource constraints (27) by some strong valid inequalities. The new constraints add finish-start precedences between trucks such that resource constraints are not violated. Note that we only need to consider sets of trucks whose execution time windows can overlap, which limit the number of critical sets.

Note also that a solution for the relaxed problem (without constraints (27)), in which valid inequalities have been added, gives a lower bound for the non-relaxed problem. Of course this solution might induce some resource capacity violations. A resource capacity violation is characterized by the fact that some critical sets do not respect the minimum number of finish-start precedence constraints that guarantees the resource feasibility (i.e., constraints (27)). Such a critical set will give rise to a new valid inequality as explained below.

Theorem 6. *Considering a single resource of capacity* n *and a critical set* $e \in E$ *of cardinality* $|e| > n$*, the following strong inequality holds:*

$$\sum_{(u,v)\in e^2, u\neq v} \alpha_{uv} \geqslant \frac{(|e|-n)(|e|-n+1)}{2}. \tag{31}$$

Note that equation (31) effectively strengthens constraints (27) as the right member can only be greater than or equal to $|e|-n$, following the definition of a critical set. Note also that these inequalities are very generic as they can be used for any scheduling problem with cumulative resource constraints, not only the door scheduling problem.

Proof. Part one: (31) *is a strong valid inequality*

Consider the set $\mathcal{U}$ of all truck intervals, and let $G(\mathcal{U})$ be the associated interval graph. Let Q be any maximal clique on $G(\mathcal{U})$, i.e., a set of mutually overlapping truck intervals, and denote $|Q| = k_Q$. If $k_Q > n$, in any feasible solution all k_Q trucks of Q cannot be scheduled in parallel, so precedences must be introduced. The question is what is the minimum number of precedences that need to be imposed. In other words, if we consider all variables α_{uv} for $u, v \in Q$, what is the minimum number of variables that have to equal 1 in any feasible solution.

In the following, all concepts refer to a subset Q of trucks. So, we refer to schedules, start times, completion times of trucks in Q. Given a feasible schedule σ, let s_u denote the start time of truck u, and let $C_u = s_u + p_u$ be the completion time of truck u. In σ, for each door $g \in [\![1, n]\!]$, there is a sequence of k_g trucks. Let (g, w) denote the w^{th} truck assigned to door g, $g \in [\![1, n]\!]$, $u \in [\![1, k_g]\!]$. We let $s^g(1), s^g(2), \ldots, s^g(k_g)$ denote the start times of the k_g trucks assigned to door g.

Given a feasible schedule σ, we say that *a precedence occurs* between trucks (g, w) and (h, v) if $s^g(w) + p_w \leqslant s^h(v)$ or $s^h(v) + p_v \leqslant s^g(w)$. Assuming the feasibility of schedule σ, we want to establish a lower bound on the number of precedences occurring between trucks in σ.

Observe that without loss of generality we can replace σ with a new (artificial) schedule σ' in which each truck has the same start time as in σ, but their processing time is *lengthened* so that its completion time coincides with the start time of the successive truck on the same door. Let us point out that this replacement cannot increase the number of precedences among trucks (possibly it can decrease as some trucks on different doors can become *overlapping*). Formally, we redefine the completion times of the trucks as $C^g(w) = s^g(w+1)$, $g \in [\![1, n]\!]$, $w \in [\![1, k_g - 1]\!]$. From now on we name this schedule σ'. Note that in σ' there are no idle times on doors, i.e., door g is busy from time $s^g(1)$ to $C^g(k_g)$. We call an "*event*" the completion of a truck (g, w) and the beginning of truck $(g, w+1)$, $g \in [\![1, n]\!]$, $w \in [\![1, k_g - 1]\!]$. We let $\tau(g, w)$ be the time at which the *event* corresponding to the com-

pletion of truck (g, w) takes place. Note that in σ' there are exactly $k_Q - n$ events.

Now let us establish the main argument. We first assume that events are totally ordered in time, i.e., no two events take place simultaneously. This assumption cannot increase the number of precedences as, in the particular case of two events taking place simultaneously, one extra precedence occurs. First of all, any single event is attached to a obvious precedence, due to directly consecutive trucks (hence corresponding to the truck before and the truck after the event), i.e., (g, w) and $(g, w+1)$ for the event $\tau(g, w)$. Since the number of events is equal to $k_Q - n$, we have a first set $k_Q - n$ precedences. Secondly, we can define additional precedences generated by pairs of events. If $\tau(g, w) \leqslant \tau(h, v)$, then a precedence occurs between trucks (g, w) and $(h, v+1)$. Every pairs of events lead to at least one precedence. Since the number of events is equal to $k_Q - n$, there are $(k_Q - n)(k_Q - n - 1)/2$ additional precedences among trucks. Hence the total minimal number of precedences is

$$\frac{(k_Q - n)(k_Q - n - 1)}{2} + (k_Q - n) = \frac{(k_Q - n)(k_Q - n + 1)}{2}. \quad (32)$$

As precised earlier, if two events take place at the same time, the number of precedences is higher than what expressed by (32). Indeed $\tau(g, w) = \tau(h, v)$, leads to two precedences instead of one: (g, w) precedes $(h, v+1)$ and (h, v) precedes $(g, w+1)$. □

The inequality being proved, we show below that it also enforces the formulation as it does not result from a linear combination of constraints (27), i.e. a combination of minimal critical set inequalities. The following part is a proof by counterexample.

Proof. Part two: (31) *is not a linear combination of the constraints.*

Consider a critical set e composed of four trucks a, b, c, d ($k_Q = 4$) and a resource limit of 2 ($n = 2$). We introduce $\alpha_{uv} = 1$ when $u \prec v$, (u, v) being any truck in $\{a, b, c, d\}$. The minimal critical set has cardinality 3 ($n + 1$). Any set composed of 3 trucks among e is a minimal critical set. The number of minimal critical set is $\binom{k_Q}{n+1} = \binom{4}{3} = 4$, namely: $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, and $\{b, c, d\}$.

With this example, constraints (27) can be applied on a total of 5 critical sets: one set of 4 trucks and four sets of 3 trucks. However, the following shows that the constraint applied on the full set e (4 trucks) is a linear combination of the constraints applied on the minimal critical sets.

We are focusing on minimal critical-set constraints. A minimal critical-set constraint imposes at least one precedence ($|e| - n = 1$) among trucks from the set. It is well known that imposing one precedence constraint for all minimal critical sets is necessary to ensure the

feasibility of the resulting schedule (i.e., there is no remaining critical set), provided that the resulting temporal graph does not contain any positive length circuit. In our example, for each minimal critical set, it is possible to choose among 6 possible precedences (for example $a \prec b$, $b \prec a$, $a \prec c$, $c \prec a$, $b \prec c$, $c \prec b$ in a, b, c). The first four following equations correspond to the inequations (27) applied to our minimal critical sets. By adding all four constraints and dividing the obtained result by 2, we obtain the constraints (27) applied to critical set e.

$$
\begin{array}{llll}
\alpha_{ab}+\alpha_{ba}+\alpha_{ac}+\alpha_{ca} & +\alpha_{bc}+\alpha_{cb} & & \geqslant 1 \\
\alpha_{ab}+\alpha_{ba} & +\alpha_{ad}+\alpha_{da} & +\alpha_{bd}+\alpha_{db} & \geqslant 1 \\
+\alpha_{ac}+\alpha_{ca} & +\alpha_{ad}+\alpha_{da} & +\alpha_{cd}+\alpha_{dc} & \geqslant 1 \\
 & +\alpha_{bc}+\alpha_{cb} & +\alpha_{bd}+\alpha_{db}+\alpha_{cd}+\alpha_{dc} & \geqslant 1 \\
\hline
\alpha_{ab}+\alpha_{ba}+\alpha_{ac}+\alpha_{ca} & +\alpha_{bc}+\alpha_{cb}+\alpha_{ad}+\alpha_{da} & +\alpha_{bd}+\alpha_{db}+\alpha_{cd}+\alpha_{dc} & \geqslant 2
\end{array}
$$

Considering the entire critical set e, applying our inequality returns a right-term of $\frac{(k_Q-n)(k_Q-n+1)}{2} = 3$, thus

$$\alpha_{ab}+\alpha_{ba}+\alpha_{ac}+\alpha_{ca} \quad +\alpha_{bc}+\alpha_{cb}+\alpha_{ad}+\alpha_{da} \quad +\alpha_{bd}+\alpha_{db}+\alpha_{cd}+\alpha_{dc} \quad \geqslant 3$$

Our inequality is therefore stronger than a linear combination of any minimal critical-set constraints. □

Figure 13 illustrates the interaction of precedences with four trucks on two doors, following the previous example. Considering a given minimal number of precedence constraints, a fesasible schedule with the exact number of precedences is proposed. We can note that 2 precedences correspond to $k_e - n$, the number of trucks exceeding the resource limit, an intuitive minimum number of precedences. We can observe critical sets in (i, ii, iii), with respectively $\{0, 1, 2\}$ precedences because the number of precedences existing between the trucks is too low. As expected, because the valid inequality requires a minimum of 3 precedences, 2 precedences are not enough to break the critical set.bIn (iv), with the 3 mentioned precedences, it is possible to respect the resource limit. A number of three precedences is the minimum required to avoid the critical state in this example.

However, as previously mentioned, this minimum number of precedences is not a sufficient condition to guarantee the feasibility of any schedule, as illustrated in Figure 15.

Nevertheless, this example tends to indicate that it is not possible to enforce even more the valid inequality to obtain a sufficient condition, as the feasible solution in Figure 13 (iv) would not be feasible anymore.

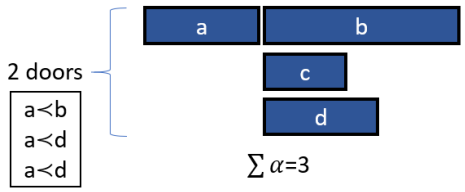

Figure 15: A necessary but not a sufficient valid inequality.

It is worth noting that the valid inequality (31) does not strengthen constraints (27) if the critical set is minimal, as in that case (27) and (31) are strictly equivalent (they both request one precedence). That is why in our approach, as detailed below, we will not restrict our attention to only minimal critical sets.

7.2.1 *Illustration*

Based on the instance introduced in chapter 3, we can consider a violated critical set in Figure 16. Considering a 2 doors crossdock, the capacity constraint is violated in three time slots. In order to solve the critical state, the valid inequality enforces some precedences. The size of the critical set is 4, while the limit of simultaneous trucks is 2. The valid inequality imposes a minimum of $\frac{(4-2)(4-2+1)}{2} = 3$ end-to-start precedences between trucks in this set.

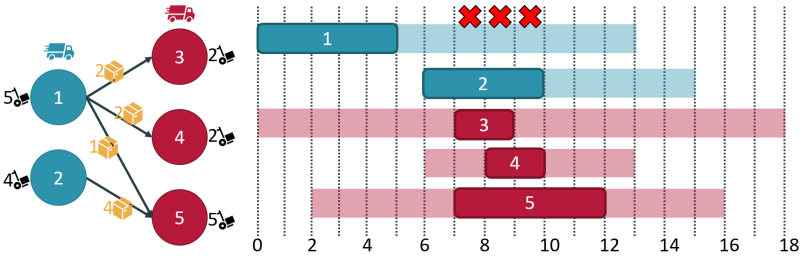

Figure 16: Violated critical set on the example

Figure 17 presents the critical set of the example, after the introduction of 3 precedences: $2 \prec 4, 3 \prec 4$ and $3 \prec 5$. The capacity constraint is now respected in all time slots. Using less precedences cannot solve the violation of critical set.

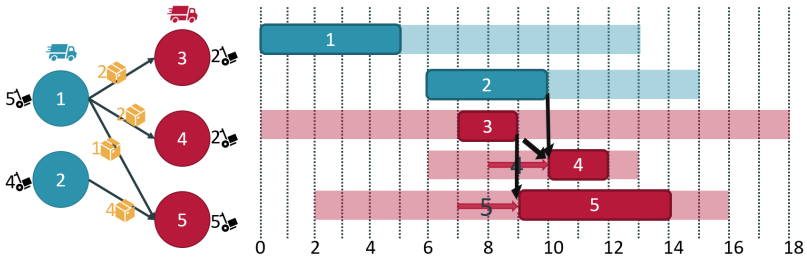

Figure 17: Resolved critical set on the example

7.3 OPTIMISATION STRATEGY

7.3.1 *Limitation of cuts application*

The valid inequalities being now defined, they can be applied on the relaxed problem. Of course they cannot be all introduced initially into the relaxed problem because of their exponential number. For that reason, this section details an approach to separate them iteratively according to the following algorithm sketch.

The algorithm below distinguishes two models: the original model, with resource constraints, and the iterative model, initially relaxed and iteratively enriched by valid inequalities. The status of a solution can be different with respect to these models. A feasible solution for the iterative model can be *infeasible* for the original model, due to relaxed constraints. Obviously, an *optimal* solution for the iterative model can either be infeasible for the original model if a resource constraint remains violated, or optimal. For clarity reasons, the solution status with respect to the iterative model is *emphasized*, whereas it will be regularly written when considering the original model.

In order to keep the methodology efficient, the number of added valid inequalities should remain as small as possible so that the computational effort needed at Step 2 to exhibit a new feasible solution remains low. But considering a non resource-feasible solution, many valid inequalities can be exhibited. Four different strategies are considered below in order to select which valid inequalities should be added in order to keep the solving methodology efficient.

7.3.2 *Cut selection strategies*

The selection of the most promising cuts is a difficult task. Indeed, even considering a single critical set, the number of existing valid inequalities can be very high (i.e., exponential with the cardinality of the critical set), as critical subsets can be considered. The cardinality of the considered critical set defines the number of imposed precedences. Considering a non-minimal set ($|e| > n + 1$), there exist critical sets composed with trucks from e. Any subset $|e_{sub}|$ from e

Algorithm 1 iterative cut algorithm

Input: instance data
Output: schedule

relaxing original problem (the resource constraints (27) are removed)
while optimal not found **do**
  solving the current model, until a *feasible* solution is found.
  if solution feasible for the original formulation and *optimal* for the current model. **then**
    global optimum obtained and the algorithm ends
  else if solution unfeasible for the original formulation. **then**
    defining some valid inequalities selected from some critical sets violated in the proposed solution.
    adding valid inequalities to the current model.
  end if
end while

with $|e_{sub}| \geqslant n+1$ is critical. Considering a maximum of $|e|$ trucks in parallel, the number of critical sets defined by these trucks are $\sum_{x\in[n+1,|e|]} \binom{|e|}{x}$, all possible sets from size $n+1$ to $|e|$.

To applied cuts, we consider a solution of the relaxed problem as well as the number of dock doors being used along the time-horizon for this solution, as illustrated in Figure 18.

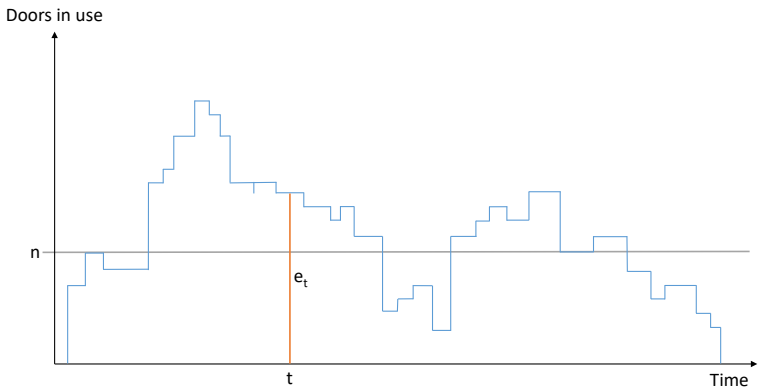

Figure 18: Doors usage on example solution

As the relaxed model does not take into account all resource constraints, the limit of n doors is exceeded during some time periods. Although the figure does not represent trucks individually, all trucks are taken into account in the obtained solution. At any time t, the set e_t of the trucks being handled is known. According to the shape of the graph and the current time t, different sets e_t (and critical subsets in e_t) can be considered.

From one solution to another, new valid cuts are added in order to progressively converge towards a resource-feasible solution for the original problem so that all critical sets have been avoided. However, to avoid handling too much cuts, these critical sets are not considered at once. We review below four cut selection strategies, namely the *top cut*, *base cut*, *k-base cut - static*, and *k-base cut - adaptive*.

TOP CUT The first strategy that we introduce is called the top cut strategy and focuses on critical sets with highest violation of the resource capacity. For each peak of the workload curve associated with the current solution, we can identify a critical set of maximum size and generate a cut. We say that a peak occurs at time t when the number of doors used at time t is strictly greater than the one used at time $t-1$, strictly greater than n, and remain constant until time $t+k(k>0)$ when it decreases again.

As the total number of peaks is relatively small, a moderate number of cuts is added at each iteration. Nevertheless, these cuts are relatively strong as the number at the right side of the inequality is high. This strategy tends to reduce the worst capacity violation from solution to solution, decreasing progressively the height of the peaks. More precisely, t being the time when a peak of resource consumption occurs, the top cut strategy applies the valid inequality on the entire set of trucks executed at time t.

Figure 19 illustrates the top cut strategy on a workload curve example. For this example, the algorithm detects 5 peaks, which will generate 5 additional valid inequalities in the model. Obviously, e being a critical set for which a cut is added, trucks of e will not be handled in parallel at the same time in the next solution, as precedence constraints between them have to be added (i.e., the same solution cannot be reached twice). However, considering a peak with a non-minimal critical set, part of the trucks belonging to the set can still form another critical set, involving also new trucks. As the number of trucks considered by the schedule is limited, peak heights will strictly decrease. Thus, this strategy can be used until a feasible solution for the initial problem is found.

BASE CUT The second strategy that we introduce is called the base cut strategy which focuses on minimal critical set having a cardinality that equals $n+1$. Among the trucks from the beginning of the resource violation to its end, a set of $n+1$ parallel trucks is selected and a cut is added into the model that forces to set up at least one precedence between them. Since a resource overrun can last a long time, there can potentially be multiple choices of the critical minimum set. In this case, we select the set of trucks having the biggest time intersection. The number of expected cuts from one solution to another is low. Obviously, limiting the number of cuts by iteration

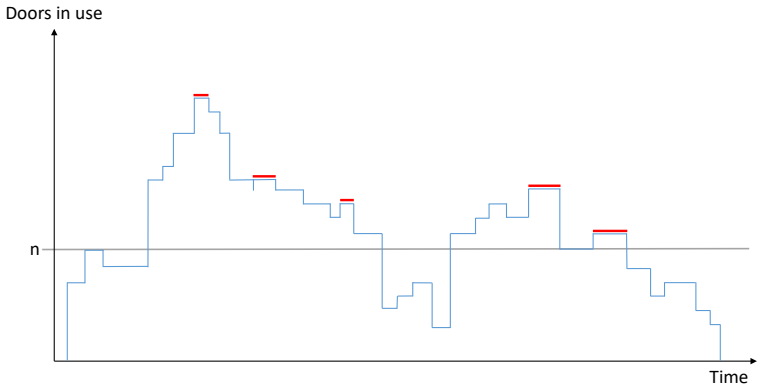

Figure 19: Illustration of the top cut strategy.

requires more iterations, but solving the model from one iteration to another may be easier.

Figure 20 illustrates the base but strategy on the same workload curve example as before. The proposed solution will generate 3 minimal cuts, and for each cut, $n+1$ trucks with largest overlap are selected among the trucks that are docked during the dark blue line. This strategy does not rely on the valid inequality, as we consider minimal cuts of $n+1$ trucks.

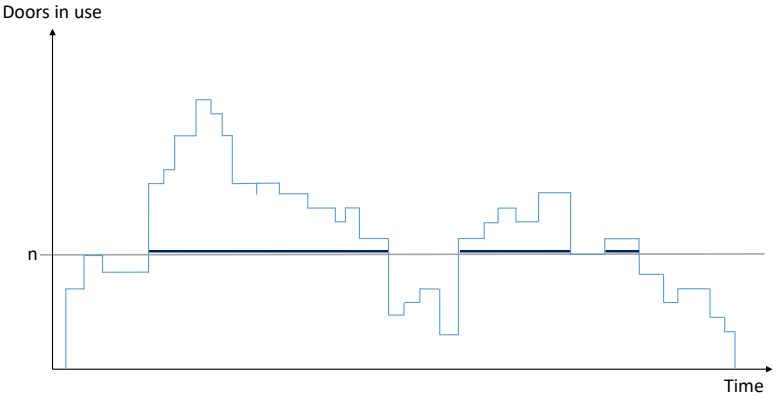

Figure 20: Illustration of a base cut strategy

κ-BASE CUT - STATIC The two previous strategies consider either a maximal set or a minimal set of trucks. Instead of using one extreme or the other, the κ-base cut is used as a trade-of approach where we select a set of $\kappa+1$ trucks, with $\kappa \geqslant n$. We propose first a *static* variant of the strategy where $\kappa_t = \mathrm{floor}(n+\lambda(k_{max}-n-1)$ with $\lambda \in [0,1]$, and k_{max} the maximum number of trucks docked simultaneously over the entire time horizon for the current solution. The selection of the critical set for the application of the cut mirrors the base cut strategy, but instead of n, the threshold limit to generate a new cut is

now fixed to κ. An intermediate value of $\lambda = 0.5$ as been selected after a short study. Similarly to the top cut strategy, we highlight that a subset of the trucks involve in the cut can still form a critical set from a solution to another. However, the cardinality of such critical sets will get lower and lower, until $\kappa = 0$ at the end of the algorithm (i.e., the solution is resource feasible). Figure 21 illustrates this strategy. The new limit κ, used to apply cuts, is materialized, and the green lines highlight the trucks which can be involved in the cuts.

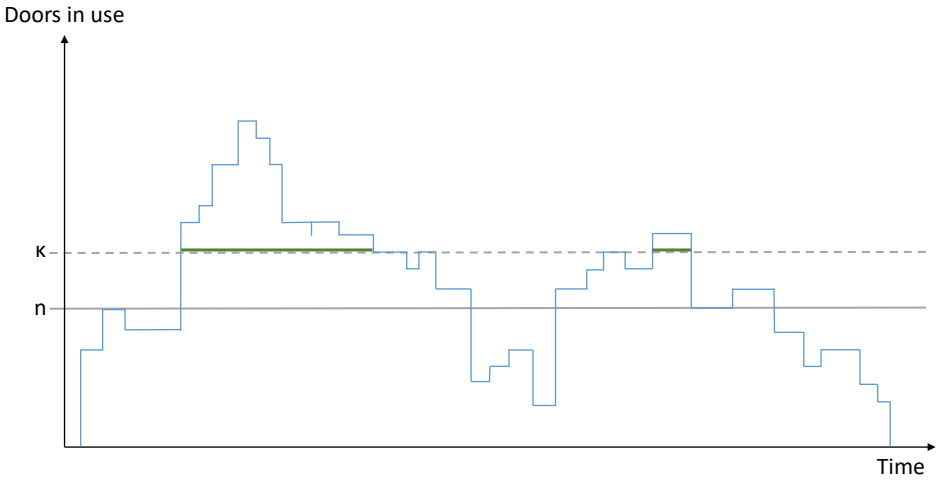

Figure 21: Illustration of κ-base cut strategy - static

κ-BASE CUT - ADAPTIVE In this second variant of the κ-base cut, an *adaptive* limit for generating cuts is chosen, meaning that the value of the threshold κ will now depend on the peaks of the workload curve. Let refer to κ_t as the resource consumption associated to a peak of the workload occurring at time t. As for the first variant, for each cut, a set of κ_t trucks is selected. The application of the cut mirrors the top cut strategy. For any peak of k_t exceeding the door capacity n ($k_t > n$), a cut is applied involving $\kappa_t + 1$ trucks, such that $\kappa_t = \text{floor}(n + \lambda(k_t - n - 1)$ with $\lambda \in [0, 1]$, among the k_t trucks of the peak. Figure 22 illustrates this strategy for the case where $\lambda = 0.5$. The limit κ_t, used to apply cuts is materialized with a light green line. Top cuts are still indicated in red.

We highlight that top cut and base cut strategies can be seen as specific cases of the adaptive κ-base cut strategy such that $\lambda = 1$ and $\lambda = 0$, respectively.

7.4 PERFORMANCES OF THE CUT STRATEGIES

The proposed algorithm is tested on the instances introduced in chapter 4. The various strategies presented in this chapter are implemented with a maximal time limit of 2 hours. The platform is

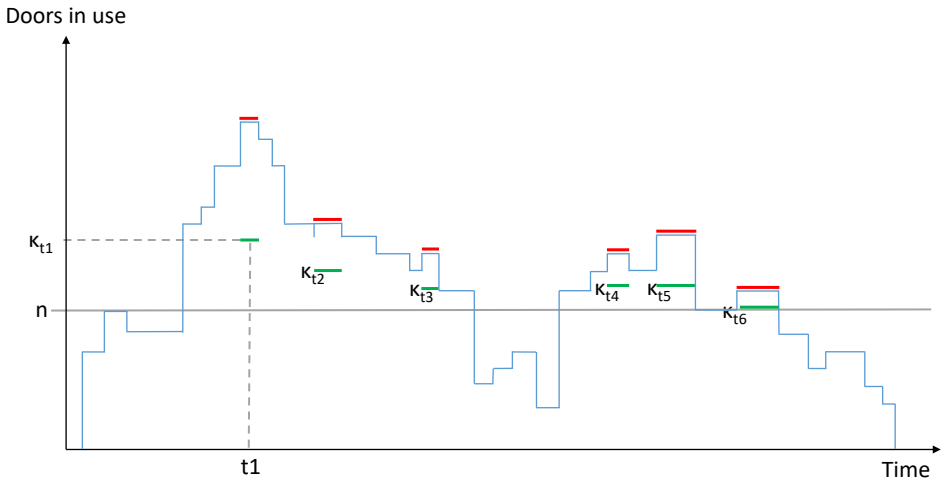

Figure 22: Illustration of κ-base cut strategy - adaptive

identical to the one used for the time indexed computation. Further comparisons and technical details will be given in chapter 9. This subsection aims to validate our solving approach and to compare our 4 cut-selection strategies.

1000 correlated instances are solved using each of the 4 strategies, then 1000 unrelated instances are solved. The following tables mention the number of optimal (*opt.*), feasible (*feas.*), proved infeasible (*infeas.*), or unknown (*unkn.*) instances. The mean gap, and the mean time (in seconds) needed to solve the instances is also reported. Results are spread according to the number of doors and according to the tightness of the time-windows. (As a reminder, "0.00" stands for the tightest time windows, while "1.00" stands for the largest time windows.)

Table 10: Correlated top cut results

		top					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		205	0	45	0	0,0%	9,5
2		31	183	35	1	47,9%	5488
5		0	104	26	120	77,1%	6454
10		0	123	16	111	83,0%	6787
	0.00	22	41	115	22	22,9%	2342
	0.25	55	73	6	66	40,9%	5088
	0.50	51	85	1	63	44,7%	5358
	0.75	54	103	0	43	48,7%	5316
	1.00	54	108	0	38	49,9%	5319
all	all	236	410	122	232	44,1%	4684

Table 11: Correlated base cut results

		base					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		205	0	45	0	0,0%	3,4
2		32	183	35	0	47,1%	5474
5		0	53	26	171	81,3%	6452
10		0	53	13	184	70,9%	6848
	0.00	22	40	112	26	25,0%	2412
	0.25	54	88	6	52	46,3%	5097
	0.50	53	50	1	96	29,1%	5324
	0.75	53	56	0	91	32,0%	5333
	1.00	55	55	0	90	32,5%	5305
all	all	237	289	119	355	34,6%	4694

Table 12: Correlated κ-base static cut results

		κ-base static					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		205	0	45	0	0,0%	4,8
2		24	190	35	1	50,4%	5593
5		0	91	26	133	80,3%	6453
10		0	111	10	127	81,7%	6941
	0.00	22	41	109	26	24,4%	2487
	0.25	54	85	6	55	46,0%	5107
	0.50	51	69	1	79	40,2%	5366
	0.75	51	94	0	55	47,1%	5368
	1.00	51	103	0	46	49,2%	5367
all	all	229	392	116	261	43,7%	4744

Table 13: Correlated κ-base adaptive cut results

		κ-base adaptative					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		205	0	45	0	0,0%	4,4
2		25	190	35	0	48,9%	5574
5		0	91	26	133	81,1%	6452
10		0	94	13	143	81,8%	6878
	0.00	22	40	112	26	24,1%	2454
	0.25	54	83	6	57	44,2%	5102
	0.50	52	63	1	84	36,1%	5343
	0.75	51	90	0	59	46,1%	5368
	1.00	51	99	0	50	49,2%	5369
all	all	230	375	119	276	42,3%	4727

Table 14: Unrelated top cut results

		top					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		201	0	49	0	0,0%	1,5
2		51	164	35	0	38,1%	4918
5		1	125	24	100	72,4%	6510
10		0	161	8	81	83,1%	6992
	0.00	24	51	105	20	20,3%	2623
	0.25	53	87	11	49	41,8%	4932
	0.50	60	88	0	52	44,0%	5130
	0.75	58	118	0	24	51,0%	5166
	1.00	58	106	0	36	47,9%	5175
all	all	253	450	116	181	43,7%	4605

Table 15: Unelated base cut results

		base					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		201	0	49	0	0,0%	1,1
2		58	157	35	0	34,9%	4735
5		1	74	24	151	75,8%	6488
10		0	68	8	174	68,7%	6980
	0.00	24	50	105	21	20,7%	2581
	0.25	54	95	11	40	45,7%	4901
	0.50	61	45	0	94	24,5%	5071
	0.75	61	52	0	87	28,6%	5097
	1.00	60	57	0	83	31,6%	5104
all	all	260	299	116	325	32,0%	4551

Table 16: Unrelated κ-base static cut results

		κ-base static					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		201	0	49	0	0,0%	1,4
2		52	163	35	0	38,7%	4850
5		1	108	23	118	73,4%	6534
10		0	135	6	109	79,4%	7033
	0.00	24	51	102	23	20,9%	2706
	0.25	53	98	11	38	45,9%	4918
	0.50	60	61	0	79	34,3%	5098
	0.75	58	94	0	48	45,4%	5154
	1.00	59	102	0	39	46,5%	5146
all	all	254	406	113	227	41,0%	4605

Table 17: Unrelated κ-base adaptive cut results

		κ-base adaptative					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		201	0	49	0	0,0%	1,4
2		55	160	35	0	37,1%	4799
5		1	105	24	120	76,2%	6494
10		0	111	7	132	79,9%	6999
	0.00	24	51	104	21	21,4%	2612
	0.25	54	90	11	45	43,7%	4899
	0.50	60	57	0	83	31,8%	5095
	0.75	59	89	0	52	44,5%	5138
	1.00	60	89	0	51	45,0%	5123
all	all	257	376	115	252	39,4%	4573

On the overall, about 25% of the instances are solved to optimality, and ~12% are proved unfeasible on the correlated and unrelated instances. The number of feasible and unknown instances have more variation according to the strategy: 23% to 35% for correlated and 18% to 32% for unrelated are still unknown after 2h of computation.

As on might expect, optimally solved instances are mainly one door instances, and a part of the two doors instances. Moreover, all one door instances without optimal solution are proved unfeasible, meaning the method solved 100% of the one door instance. The computation of one door instances are also really quick, some seconds are needed on average.

The critical sets method begins to struggle with two doors instances, with ~75% (~64%) of the correlated (unrelated) instances being still under computation after reaching the time limit. However, except from one unknown instance, a solution is always available, with an average gap of ~50% (~37%) for correlated (unrelated) instances. Nevertheless the method completely solves ~25% (~30%) of the correlated (unrelated) two doors instances. Focusing on 5 and 10 doors, no optimal solutions are found anymore, and less instances are proved unfeasible with a sensitive difference between 5 and 10 doors. Although the slight differences in number of unknown instances, the increased difficulty to solve 10 doors instances is reflected by the average gap and computation time.

Looking at the time windows, an overall analysis can suppose a reduced difficulty on the tightest instances (0.00): the number of optimal solutions found is lower, but undoubtedly the number of unfeasible instances is higher due to the reduced time windows. In the correlated set 0.00, the unfeasible instances are at least 115 (top cut infeasible) and at most 135 (κ-base cut unfeasible + unknown). In correlated set 0.25, the unfeasible instances are at least 6 (top cut infeasible) and at most 58 (base cut unfeasible + unknown). Very similar

values are obtained for unrelated sets. Nevertheless, for the 0.00 set, the method is able to prove unfeasible a large part of the instances, and only a few ones (~10% stay unknown), while the average gap and solving time are substantially reduced compared to other sets (0.25, 0.50, 0.75 and 1.00).

The efficiency of the method will be further developed, with comparison to other methods in chapter 9. The choice of the best cut strategy is discussed below.

κ-BASE STRATEGY First of all, the κ-base strategy presents very similar results for both variants. On the correlated and unrelated cases, a single relevant difference relies on the unknown (and indirectly feasible) number with 10 doors, where the static variant appears less strong. The adaptive variant has been selected for comparison with top and base strategy.

OVERALL STRENGTH OF TOP CUT STRATEGY Looking at the overall results, the top cut strategy presents the most interesting solutions. The number of optimal solutions and infeasible instances is similar to the other strategies, whereas the base cut strategy presents a slight upgrade. However, the ability to find solutions is highlighted when looking at instances recognized as unknown. Relatively to top cut, for base cut, the number of unknown increases by 53% and 80% for correlated and unrelated instances respectively. Relatively to top cut, for the κ-base static cut, the number of unknown increases by 12% and 25% respectively.

To conclude, the top cut strategy is selected as the most promising for this resolution method. Although the top cut does not dominate the base cut, in term of optimally solved instances, the better performance is clearly visible on unknown instance. Both base and κ-base lack of efficiency to obtain feasible solutions.

8

CONSTRAINT PROGRAMMING

In this section, we propose to address the CTSP by constraint programming (CP) as an alternative solving methodology. CP considers a set of variables having their specific domain of values (either binary, discrete or continuous) and a set of constraints that link the variables together. Constraints can be logical or arithmetic (e.g, different, equal, less-or-equal, etc.) or more global (e.g., all-different, knapsack constraint, 1-among-n constraint, cumulative constraint, etc.). Given a constraint corpus, a set of constraint propagation procedures is defined that allows, considering different pre-assignments of variables, to reduce the domains of variables by identifying impossible configurations. This process is particularly efficient in finding feasible solutions as constraint propagation mechanisms, which are very effective in practice (their worst computation complexity being polynomial), can be advantageously combined with other decision procedures assigning values to variables in an iterative way. CP is an interesting paradigm now able to compete with other Linear and Integer Programming approaches (LP/IP) and is often used in literature to efficiently solve scheduling problems (Le Pape, 2014; Zeballos, 2010).

The literature using CP for solving the CTSP is scarce. To the best of our knowledge, only Fazel Zarandi et al. (2016) propose to use CP for solving a just-in-time CTSP with preemption.

8.1 A CONSTRAINT-PROGRAMMING MODEL FOR THE CTSP PROBLEM

To solve one decision or optimization problem, CP allows various modelings that differ according to the chosen solver and the implemented constraints. In Chapter 5 of their book, Bourreau et al. (2019) recommend a specific modeling of resource limitation in scheduling problems that use the powerful IBM - CP Optimizer (CPO) solver, relying on the concept of time interval decision variables.

Therefore, to take benefit from this interesting concept and its associated constraint propagation procedures, we choose to use CPO in our work. Time interval variables have been associated to trucks, each variable having two attributes, *size* and *start*, representing the docking duration and the starting time of handling, respectively. Additionally, a global constraint is used in symbiosis with interval variables to control the resource usage, i.e. the doors are considered as a single cumulative resource available in multiple units. A *Cumul Function* in CPO allows to count overlapping intervals over time, and a single con-

straint $\mathrm{Cumul\_Function} \leqslant n$ is enough to impose that the resource consumption should not exceed the resource capacity. Thanks to this concept, CPO allows a compact and straightforward CP fomulation, as presented in the following.

The decision variables are the starting times of the intervals $\mathcal{S}_i$ ($\mathcal{S}_o$) which represent the duration of the unloading operation of an inbound truck $i \in I$ (the duration of the loading operation of an outbound truck $o \in O$). This duration, the size of the interval, is by default a variable, but in our problem, it is fixed and dependant of the truck.

A CP model for the considered CTSP problem is the following:

$$\min \quad z = \sum_{(i,o)\in A} w_{io} \left(\mathrm{start}(\mathcal{S}_o) - \mathrm{start}(\mathcal{S}_i)\right) \tag{33}$$

subject to

$$\mathrm{size}(\mathcal{S}_u) = p_u \qquad \forall u \in U \tag{34}$$

$$\mathrm{start}(\mathcal{S}_i) - \mathrm{start}(\mathcal{S}_o) \leqslant 0 \qquad \forall (i,o) \in A \tag{35}$$

$$\mathrm{Cumul\_Function}(\mathcal{S}_u) \leqslant n \tag{36}$$

The objective function (33) is similar to the previous critical-set formulation. Constraints (34) force the interval size of each truck to have the length of the processing time. Constraints (35) enforce the start-start precedence constraints between any pair of inbound truck i and outbound truck o such that $(i,o) \in A$. The global constraint (36) imposes that the number of trucks simultaneously docked cannot exceed the total number of doors n.

In the critical-set formulation of chapter 7, the limited capacity of the door resource is modelized with precedences between trucks belonging to same critical sets. With CP, the cumulative resource constraint is expressed more straightforwardly and his treatment is totally integrated inside the various propagation algorithms. It makes the solving process very effective, as proved by our experiments.

8.2 ITERATIVE CUTS ON CP

The straightforward CP formulation bringing interesting results, we attempt to mix our critical-set formulation presented in the previous chapter 7 with the CP formulation. This section details a first way to do it.

In order to integrate the valid inequality into the CP model, we need to modify the resource constraint since one cannot express directly the valid inequality as a global CP constraint. Therefore, the cumulative constraint (36) was replaced by constraints based on precedences and critical sets. Our motivation is to determine whether

adding valid inequalities iteratively can compensate for the loss of not using the cumulative global constraint. To use the valid inequality within the CP formulation, binary variables α_{uv} were added to model precedence constraints between pair of trucks. Next, CPO allows to achieve reification that consists in activating the end-to-start CPO precedence constraint $u \prec v$ if and only if binary variable α_{uv} is set to one. For that reason, each valid inequality defined in chapter 7 gives rise to a CPO constraint, which requires that a minimum number of precedence variables are set to one.

Given a critical set $e \in E$, the valid inequality is expressed as follows:

$$\sum_{(u,v)\in e^2, u\neq v} \alpha_{uv} \geqslant K \tag{37}$$

α_{uv} is the binary precedence variables, and K is the value returned by the valid inequality. Note also that the model integrates the constraints $\alpha_{uv} + \alpha_{vu} \leqslant 1, \forall (u,v) \in U^2, u \neq v$.

As mentioned earlier, incorporating our valid inequality into the CP model may potentially induce a performance loss since CPO's cumulative resource constraint modeling, which is powerful in practice, cannot be used anymore.

8.3 HYBRIDIZATION AND SOLUTION REPAIRING

This section investigates a second way to integrate our critical-set and CP formulations in an hybrid algorithm. Instead of trying to incorporate the cuts into CP, we alternatively aim to integrate CP in our critical-set method in order to quickly find feasible solutions (as it is a well-known advantage of CP).

The method still relies on the critical-set based algorithm presented in chapter 7. The main change lies in the iterative use of CP to speed up the search for feasible solutions after the addition of a new valid inequality. The global cumulative constraint is used for this search. The basic idea is to take advantage of both the MIP aproach (to exhibit good quality lower bounds), and the straightfoward CP (to find good quality upper bounds). We further refer to this method as *CP-hybrid*.

The diagram in Figure 23 represents the algorithm and highlights the CP interaction. Colors are used to help the understanding: blue refers to the initial critical-set model, red to the model.

Let remind that after adding a cut, the last solution found is not feasible anymore. To pursue the algorithm, a new feasible solution, satisfying the new added cut has to be found. While the initial method uses the MIP solver to find a new feasible solution, starting from the last one, we now use CP for this purpose. The straightforward CP model is used to repair the solution found by the MIP model in order to satisfy the cumulative resource constraint. Once CP has found

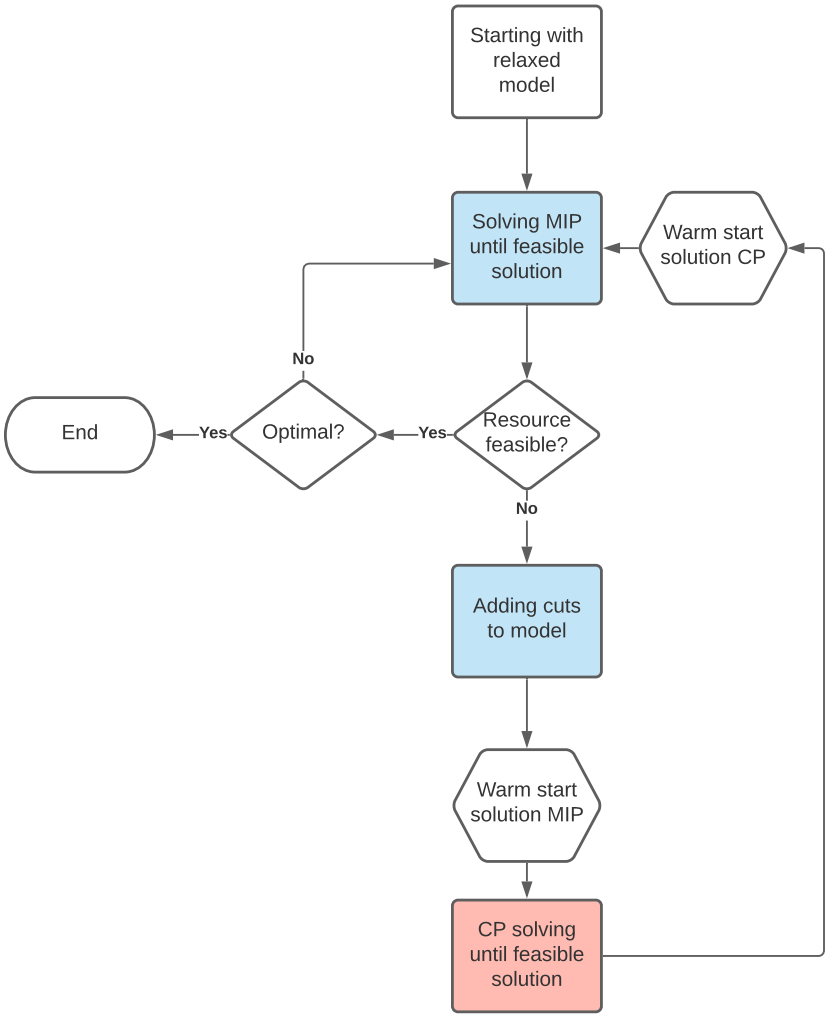

Figure 23: Hybrid algorithm

a new solution, the initial algorithm continues, using the new solution as a warm start within the MIP model. Indeed, the CP solution can potentially be improved by the MIP solver with respect to its constraints (as the resource constraints are relaxed). The best feasible solution known for the original model is saved continuously. If the solution found by CP is not good enough, the algorithm can use the saved solution instead. Moreover, to avoid spending too much time by CP to repair a solution without guarantee on improving its quality, a time limit is added. If reached, the CP stops and the algorithm continues with the MIP, from the saved solution.

8.4 PERFORMANCES AND LIMITS OF THE CP FORMULATIONS

The computational results differentiate between instances according to the number of doors and the tightness of the time windows. We reuse the 1000 correlated and 1000 unrelated instances defined in chapter 4 for this computational experiments. For each class of instances, the number of instances solved to optimality (*opt.*) is reported. If an optimal solution is not found, we count the numbers of instances for which a feasible solution is found (*feas.*) or, possibly, the number of infeasible instances (*infeas.*). Eventually, the number of instances having still an unknown status after the time limit (2h) is reported (*unkn.*). The average gap between the lower and upper bound and the mean time are also indicated in the tables.

In the sequel, we report only the performances of the straightforward CP model, followed by the ones of the CP-hybrid models. We do not comment on the results for the CP-Cut model because, as one might expect, this approach is systematically worse than the other two due to the fact that the global cumulative constraint cannot be used. The results obtained are detailed in tables 18 and 19.

Table 18: Correlated CP results

		CP					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		205	0	45	0	0%	28,2
2		83	132	35	0	60%	4070,7
5		1	222	27	0	90%	6408,8
10		0	233	16	1	88%	6739,2
	0.00	28	55	116	1	25%	2079,3
	0.25	72	122	6	0	53%	4478,4
	0.50	65	134	1	0	67%	4912,3
	0.75	62	138	0	0	69%	5038,5
	1.00	62	138	0	0	69%	5050,0
all	all	289	587	123	1	58%	4311,7

Table 19: Unrelated CP results

		CP					
n	TW	opt.	feas.	infeas.	unkn.	mean gap	mean time
1		201	0	49	0	0%	11,2
2		96	119	35	0	54%	3746,0
5		4	222	24	0	87%	6437,2
10		0	239	9	2	85%	6940,8
	0.00	31	61	106	2	20%	2325,9
	0.25	73	116	11	0	50%	4289,1
	0.50	69	131	0	0	66%	4831,8
	0.75	65	135	0	0	68%	4979,5
	1.00	63	137	0	0	69%	4992,8
all	all	301	580	117	2	55%	4283,8

First we observe that the structure of the processing times does not really influence the performances as correlated and unrelated results are highly similar. The computation time is high for almost all the instances except the ones with a single-door and the ones with tight time windows. Those instances are possibly easier because of the weaker combinatorial nature of the search space.

The efficiency of the straightforward CP model to find solution is undeniable, with an extremely low number of unknown instances. With one door, all the instances are optimally solved within a short computation time. With two doors, the ability to find optimal solutions decreases by ~40%. Moreover, CP struggles to find optimal solutions. Concerning time windows, it is hard to comment on the effect of 0.00 time windows due to the higher number of unfeasible instances, but it seems with respect to the mean gap and mean time that CP works better on tight instances. Above 0.25, no significant differences appear.

Results for hybrid-CP are presented in tables 20 and 21. We add two columns in the tables: *LB diff* and *UB diff*. The first one compares the lower bounds of the hybrid and straightforward CP in the following way: $\textit{LB diff} = \frac{LB_{hyb.} - LB_{straight.}}{max(UB_{hyb.}; UB_{straight.})}$. A positive value is hence obtained for a higher LB for CP hybrid and vice versa. Similarly, $\textit{UB diff} = \frac{UB_{hyb.} - UB_{straight.}}{max(UB_{hyb.}; UB_{straight.})}$. Remark that *LB diff* is calculated with the UB in order to have a stable reference, which cannot be obtain with LB as it is sometimes very low or zero.

The comparison shows that hybrid-CP is efficient with a number of (feasible or optimal) solutions found very similar to straightforward-CP. Moreover, hybrid-CP is faster on the one door subset, where both formulations obtain the same results in terms of lower and upper bounds. The gap appears to be slightly better on most other subsets of instances. As observed in column LB diff, this improvement can be

Table 20: Correlated CP hybrid results

		CP_hyb							
n	TW	opt.	feas.	infeas.	unkn.	mean gap	UB diff	LB diff	mean time
1		205	0	45	0	0%	0%	0%	10,6
2		33	182	35	0	48%	-9%	15%	5407,7
5		0	223	26	1	82%	-38%	16%	6453,0
10		0	233	16	1	86%	-40%	6%	6785,6
	0.00	22	61	115	2	29%	-8%	2%	2336,6
	0.25	55	139	6	0	55%	-18%	3%	5068,6
	0.50	53	146	1	0	60%	-27%	11%	5302,4
	0.75	54	146	0	0	59%	-27%	14%	5310,7
	1.00	54	146	0	0	58%	-25%	14%	5302,7
all	all	238	638	122	2	53%	-21%	9%	4664,2

Table 21: Unrelated CP hybrid results

		CP_hyb							
n	TW	opt.	feas.	infeas.	unkn.	mean gap	UB diff	LB diff	mean time
1		201	0	49	0	0%	0%	0%	1,8
2		57	158	35	0	37%	-8%	20%	4787,6
5		1	225	24	0	79%	-39%	18%	6499,1
10		0	239	8	3	85%	-43%	7%	6992,2
	0.00	24	68	105	3	26%	-11%	2%	2611,9
	0.25	54	135	11	0	52%	-21%	4%	4925,4
	0.50	61	139	0	0	57%	-27%	13%	5075,4
	0.75	60	140	0	0	56%	-27%	16%	5121,7
	1.00	60	140	0	0	56%	-26%	16%	5116,3
all	all	259	622	116	3	50%	-23%	11%	4570,1

explained by better Lower Bounds (LB), but this gain is unfortunately counterbalanced by a loss on the quality of the upper bounds (from 0% to 40%. Thus the experience of the association of MIP with CP works but remains mitigated. It seems that finding a good balance between the CPU time allocated to CP and that allocated to the MIP approach is tricky, which does not allow to take full advantage of the potential of CP to find good quality solutions.

To conclude this section, we can say that although upgrades of the basic CP formulation were tested, the results are not in favor of more complex methods. The straightforward formulation is efficient to find solutions, and using the valid inequality directly inside the model was a disappointing idea. The difficulty to improve the *basic* method may also come from the selected CP Solver. CP Optimizer is developed over years and integrates numerous optimization algorithms, beyond the scope of pure CP. However, hybrid-CP at least improves the LB, in order to better estimate the quality of the solution, without reference. Straightforward CP is selected as the best considered

method using CP, due to the proved quality of the obtained solutions. In the sequel, we will simply referred to this method as CP.

9 COMPUTATIONAL RESULTS

In this section we present the computational experiments to evaluate and compare the performance of the proposed solving methods. All experiments using a solver were running IBM ILOG CPLEX STUDIO 12.8, on Intel Xeon E5-2695 v4, 2.1GHz platform. Each instance run on a single core, with 16GB RAM available.

We built up two distinct testbeds of instances.

FIRST TESTBED This testbed, is the one already used in previous sections (see chapter 4), which contains 1000 instances with 1 to 10 doors. These instances were used to compare the different solving methods according various criteria.

SECOND TESTBED An extension of the first testbed was considered, due to the particularities of the quadratic method, which only works on single door instances without time windows. As there are only 50 relevant instances in the first testbed, all efficiently solved by the quadratic method, 110 larger single door instances were added in order to better analyse the method limits. These instances have a number of trucks $k = \{6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 28, 32, 36, 40, 44, 48\}$. The other parameters are similar to the first testbed;

The time limit for these instances is fixed to 2 hours, as determined in chapter 4.

In the following tables, we use different indicators to measure the performances of the different algorithms. For any subset of instances the solution status is reported as before. Moreover the average gap of instances with solutions is calculated in column "*Gap*" according to the formula $\frac{UB-LB}{UB}$. To realize a fairer comparison, a restrained average gap is calculated, relatively to another method M, provided that the gap is only calculated for instances with solutions found by both methods. This restrained gap is reported in column "Gap_M". For example, the column Gap_{CS} for TI results, only considers those instances for which an optimal or feasible solution was found by both TI and CS method. Therefore, Gap_{CS} of TI results and Gap_{TI} of CS results are calculated on the same sets of instances. Not considering this constrained gap can penalize a method able to find many feasible solutions on hard instances, with large gap, compared to another method returning an unknown status on these instances.

The average computation time is reported in column "*Time*".

Additionally, for any instance for which a solution was found by two methods, it is possible to calculate the difference relatively to the best solution:

$$UB\ diff = \frac{(UB_2 - UB_1)}{\min\{UB_1, UB_2\}}$$

. The average over all instances of the subset with solutions found by both selected methods is reported in column "UB_{diff}". This difference is given as a percentage, and positive if the considered method is better.

Note that the "UB_{diff}" is really useful, as one method can be more efficient on finding solutions (UB) than another, but less efficient at finding lower bounds (LB). We will see that in some cases a method can find optimal solutions, but cannot prove the optimality because it fails to find good LBs, which results in a bad gap.

9.1 FIRST TESTBED: TI, CS AND CP COMPARISON

Tables 22 and 23 present computational results, for three different groupings of the first testbed: instances are grouped based on the number of doors, the number of trucks per door and the time window tightness. The last line of the table includes the whole testbed.

First we analysis the correlated instance.

OVERALL Looking at the global results, we can observe strengths for every method. CP seems stronger than the other methods, dominating TI, and better than CS on any point except the gap. CS is more efficient than TI to find optimal solutions, and finds smaller gaps than the other methods. The smaller gap of CS is explained by better lower bounds: CS is better in estimating the solution quality, although solutions found are worse than those found by TI and CP ($UB_{diff} = -13\%$ and -14%). However, CP has difficulties to solve numerous instances (232 unknown status among 1000 instances), while TI and CP have respectively only 1 and 2 unknown solutions. We can observe that all methods are equivalent on proving infeasibily within the computation time limit.

DOOR SUBSET Focusing on subsets with 1, 2, 5 and 10 doors, each ones containing 250 instances, different behaviors are observed. CS and CP perform very well on 1 door instances, and found all optimal solutions very quickly (CS 9.5s and CP 28.2s on average). However, TI identifies all the infeasible instances, it is not able to find half of the optimal solutions. The average gap of the 205 instances with solutions (including 93 optimal with gap = 0%) is 40%. For 2 doors, CP obtains better results than the other methods, except on LB. The efficiency of CS drastically drops with increasing number of doors, and becomes even worse than TI, also for the solution quality (-5%). Nevertheless,

			TI					CS					CP				
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Time	Opt.	Feas.	Infeas.	Unkn.	Time	Opt.	Feas.	Infeas.	Unkn.	Time
1			93	112	45	0	3671,8	205	0	45	0	9,5	205	0	45	0	28,2
2			42	173	35	0	5090,7	31	183	35	1	5487,9	83	132	35	0	4070,7
5			24	199	27	0	5767,7	0	104	26	120	6453,6	1	222	27	0	6408,8
10			30	201	17	2	5908,9	0	123	16	111	6786,8	0	233	16	1	6739,2
	6		61	112	27	0	4411,8	62	107	27	4	4209,2	82	91	27	0	3331,3
	8		39	133	28	0	4969,1	45	101	27	27	4635,5	65	107	27	1	4063,2
	10		28	144	28	0	5299,4	43	89	28	40	4693,4	49	123	28	0	4479,0
	12		32	146	21	1	5372,6	43	58	21	78	4904,8	48	131	21	0	4791,6
	14		29	150	20	1	5496,0	43	55	19	83	4979,2	45	135	20	0	4893,5
		0.00	80	3	117	0	209,6	22	41	115	22	2341,6	28	55	116	1	2079,3
		0.25	71	123	6	0	4637,6	55	73	6	66	5088,1	72	122	6	0	4478,4
		0.50	20	177	1	2	6592,3	51	85	1	63	5357,7	65	134	1	0	4912,3
		0.75	9	191	0	0	7053,9	54	103	0	43	5315,8	62	138	0	0	5038,5
		1.00	9	191	0	0	7055,5	54	108	0	38	5319,1	62	138	0	0	5050,0
all	all	all	189	685	124	2	5109,8	236	410	122	232	4684,4	289	587	123	1	4311,7

			TI					CS					CP				
n	$\frac{k}{n}$	TW	Gap	Gap_{CS}	Gap_{CP}	UB_{diff} CS	UB_{diff} CP	Gap	Gap_{TI}	Gap_{CP}	UB_{diff} Ti	UB_{diff} CP	Gap	Gap_{TI}	Gap_{CS}	UB_{diff} Ti	UB_{diff} CS
1			40%	40%	40%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2			67%	68%	67%	5%	-2%	48%	48%	48%	-5%	-6%	60%	60%	60%	2%	6%
5			70%	73%	70%	26%	-5%	77%	77%	77%	-26%	-29%	90%	90%	88%	5%	29%
10			67%	69%	67%	37%	-6%	83%	83%	83%	-37%	-40%	88%	88%	86%	6%	40%
	6		51%	52%	51%	13%	-2%	43%	43%	43%	-13%	-15%	47%	47%	46%	2%	15%
	8		60%	62%	60%	14%	-2%	48%	48%	48%	-14%	-15%	58%	58%	52%	2%	15%
	10		65%	66%	65%	15%	-3%	49%	49%	49%	-15%	-17%	66%	66%	58%	3%	17%
	12		65%	61%	65%	8%	-5%	39%	39%	39%	-8%	-10%	66%	66%	48%	5%	10%
	14		66%	61%	66%	12%	-4%	39%	39%	39%	-12%	-13%	67%	67%	46%	4%	13%
		0.00	0%	0%	0%	6%	1%	23%	23%	23%	-6%	-6%	25%	25%	20%	-1%	6%
		0.25	8%	6%	8%	11%	2%	41%	41%	41%	-11%	-11%	53%	53%	37%	-2%	11%
		0.50	82%	73%	82%	11%	-6%	45%	45%	45%	-11%	-14%	67%	67%	52%	6%	14%
		0.75	90%	87%	90%	15%	-5%	49%	49%	49%	-15%	-17%	69%	69%	61%	5%	17%
		1.00	90%	88%	90%	16%	-5%	50%	50%	50%	-16%	-18%	69%	69%	62%	5%	18%
all	all	all	62%	60%	62%	13%	-3%	44%	44%	44%	-13%	-14%	61%	61%	50%	3%	14%

Table 22: Correlated results: doors, trucks per door and time windows subsets

			TI					CS					CP				
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Time	Opt.	Feas.	Infeas.	Unkn.	Time	Opt.	Feas.	Infeas.	Unkn.	Time
1			97	104	49	0	3428,3	201	0	49	0	1,5	201	0	49	0	11,2
2			45	170	35	0	5092,6	51	164	35	0	4917,5	96	119	35	0	3746,0
5			25	201	24	0	5814,0	1	125	24	100	6509,9	4	222	24	0	6437,2
10			36	202	11	1	5964,2	0	161	8	81	6992,0	0	239	9	2	6940,8
	6		60	112	28	0	4288,2	76	94	28	2	3688,7	82	90	28	0	3273,4
	8		42	130	28	0	4931,5	44	110	28	18	4666,1	68	104	28	0	4052,5
	10		33	139	28	0	5196,8	43	104	25	28	4752,2	52	120	26	2	4449,8
	12		36	147	17	0	5406,6	45	78	17	60	4971,9	51	132	17	0	4791,7
	14		32	149	18	1	5550,7	45	64	18	73	4947,3	48	134	18	0	4851,7
		0.00	88	4	108	0	316,0	24	51	105	20	2623,1	31	61	106	2	2325,9
		0.25	69	120	11	0	4616,0	53	87	11	49	4932,1	73	116	11	0	4289,1
		0.50	23	177	0	0	6530,4	60	88	0	52	5129,5	69	131	0	0	4831,8
		0.75	14	186	0	0	6886,6	58	118	0	24	5166,1	65	135	0	0	4979,5
		1.00	9	190	0	1	7024,9	58	106	0	36	5175,5	63	137	0	0	4992,8
all	all	all	203	677	119	1	5074,8	253	450	116	181	4605,2	301	580	117	2	4283,8

			TI					CS					CP				
n	$\frac{k}{n}$	TW	Gap	Gap_{CS}	Gap_{CP}	UB_{diff} CS	UB_{diff} CP	Gap	Gap_{TI}	Gap_{CP}	UB_{diff} Ti	UB_{diff} CP	Gap	Gap_{TI}	Gap_{CS}	UB_{diff} Ti	UB_{diff} CS
1			35%	35%	35%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
2			67%	67%	67%	4%	-3%	38%	38%	38%	-4%	-6%	54%	54%	54%	3%	6%
5			69%	66%	69%	28%	-2%	72%	72%	72%	-28%	-28%	87%	87%	83%	2%	28%
10			66%	71%	66%	41%	-3%	83%	83%	83%	-41%	-42%	85%	85%	85%	3%	42%
	6		52%	53%	52%	15%	-1%	40%	40%	40%	-15%	-16%	46%	46%	46%	1%	16%
	8		57%	58%	57%	15%	-1%	46%	46%	46%	-15%	-16%	54%	54%	50%	1%	16%
	10		63%	63%	63%	18%	-2%	49%	49%	49%	-18%	-19%	63%	63%	59%	2%	19%
	12		62%	60%	62%	14%	-3%	43%	43%	43%	-14%	-16%	64%	64%	51%	3%	16%
	14		66%	61%	66%	15%	-2%	42%	42%	42%	-15%	-16%	65%	65%	48%	2%	16%
		0.00	0%	0%	0%	8%	1%	20%	20%	20%	-8%	-7%	20%	20%	16%	-1%	7%
		0.25	10%	9%	10%	15%	2%	42%	42%	42%	-15%	-14%	50%	50%	39%	-2%	14%
		0.50	80%	73%	80%	13%	-4%	44%	44%	44%	-13%	-16%	66%	66%	53%	4%	16%
		0.75	87%	86%	87%	19%	-3%	51%	51%	51%	-19%	-21%	68%	68%	63%	3%	21%
		1.00	89%	86%	89%	18%	-3%	48%	48%	48%	-18%	-20%	69%	68%	62%	3%	20%
all	all	all	60%	59%	60%	15%	-2%	44%	44%	44%	-15%	-17%	59%	59%	51%	2%	17%

Table 23: Unrelated results: doors, trucks per door and time windows subsets

CS keeps its ability to find better LB. Instances with 5 and 10 doors bring unexpected results: although CS and CP showed efficiency on finding optimal solutions for less doors, TI is almost the only one able to find optimal solutions on these subsets, with almost no unknown. TI also has the best gap, with a relatively good UB (-6% relative to CP).

TRUCKS PER DOOR SUBSET A focus on the number of trucks per door subsets shows a domination of CP on any subset, except for the gap due to a better LB obtained by CS. Overall, CS finds more optimal solutions than TI, but also more unknown. When the number of trucks per door increases, the number of optimal solutions found by TI and CP gradually decreases, while the number of unknown from CS increases. This decreasing/increasing slows down with more trucks per door.

TW SUBSET Focusing on the time windows tightness highlights two phenomenons. CS and CP have a similar behavior: the results are relatively stable with the increase of the length of the time windows, with the CP dominating again CS, except on the gap due to the better LB found by CS. TI is really strong on tight TW instances while being less efficient on wider ones. For 0.00, TI found almost all optimal solutions (80 on 83), while CP found only 28. For 0.25, TI found as many optimal solution as CP, but with a far better gap (8% vs 53%). An important observation is that the UB difference between TI and CP on subset 0.00 is very low (1%). This means that CP was able to find almost all optimal solutions without being able to prove the optimality. For the 0.25 subset, the gaps are very different, but the UB difference is only 2%, implying a huge difference on the LB.

UNRELATED RESULTS A look at the results for the unrelated instances does not show any different behaviour. Values slightly differ, but are similar on many points. The whole analysis realised on correlated instances applies on unrelated ones.

To synthesise the analysis of this first table:

1. CP is stronger on low door number
2. TI is stronger on 5-10 doors
3. CP is stronger on any trucks per doors subsets
4. TI is stronger on 0.00 and 0.25 TW
5. CP is strong on any TW subset, but lack on LB calculation
6. CS returns good LB on any subsets

Tables 24 and 25 present the same computation results, but focusing on the effect of the Time Windows tightness. Instances are

grouped according to doors number with time windows, and trucks per door with time windows. First we analysis the correlated instances.

TI STRENGTH The TI method is efficient for few doors and tight TW. TI also reacts better when there are only few truck per door. The detailed results on subset 0.00 confirm the observations: from the 200 *0.00* instances, all are proved infeasible (117) or optimally solved (80) except three instances. Focusing on the *0.25* subset, the method is still strong on 1 door instances, and still finds half of the optimal solutions on 2 doors instances. For more than 2 doors, the efficiency drops with only 1 instance out of 100 with an optimal solution. The performances of TI relative to the number of trucks per door is less impacted: we notice a smoother decrease. Finally, on wide TW, the only instances optimally solved have 1 door and 6 trucks. For *0.00* and *0.25* TW, TI performs well compared to the other methods. However, for wider time windows, it cannot stand the comparison, even when the number of doors or trucks per door is small. We can see that the efficiency of the formulation depends mostly on the TW parameters, then the number of doors and finally the number of trucks per door.

CS STRENGTH The previous analysis revealed that CS has mitigated results compared with TI, and is globally dominated by CP. Nevertheless, the ability to find a relevant LB as been identified. The gap decreases while TW becomes tighter, increases much with the number of doors, but stays stable according to the number of trucks per door. The CS gap of any subset is inferior to the CP gap, except on rare instances, where it is sightly superior, due to an important UB difference in favor of CP.

CP STRENGTH Looking at the *0.00* TW subset, CP is as efficient as TI for 1 and 2 doors, and far less efficient for 5 and 10 doors, with full feasible for CP instead of full optimal for TI. However, as specified previously, the $\mathtt{UB}_{\mathtt{diff}}$ between CP and TI for the *0.00* subset is really weak (1%), meaning the feasible solutions found by CP are almost all optimal, but CP was not able to prove optimality on 5 and 10 doors. This shows the weakness of CP on finding LB when the number of doors increases. Focusing on all TW and 1 door, CP perfectly solves the instances. Increasing the number of doors results in a drop of the performance, with a frightening gap for 5 and 10 doors when $TW \geqslant 0.50$. A gap of 100% means that no LB was found, so we do not have any information on the solution quality. This analysis shows the difficulty for CP to prove optimally and find LB when there are more than 2 doors. Without knowing the optimal solutions, it is difficult to evaluate the UB values. However, precedent analysis with $\mathtt{UB}_{\mathtt{diff}}$ showed that CP is ahead of other methods, although the obtained gap

TI		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		9	41	0	0	64,78%	9	41	0	0	63,57%	20	29	1	0	35,54%	44	1	5	0	0,09%	11	0	39	0	0,00%
2		0	50	0	0	97,17%	0	50	0	0	96,77%	0	50	0	0	91,39%	26	23	1	0	4,93%	16	0	34	0	0,00%
5		0	50	0	0	99,96%	0	50	0	0	100,00%	0	50	0	0	98,98%	1	49	0	0	12,31%	23	0	27	0	0,00%
10		0	50	0	0	100,00%	0	50	0	0	100,00%	0	48	0	2	100,00%	0	50	0	0	14,37%	30	3	17	0	0,01%
	6	9	31	0	0	73,59%	9	31	0	0	72,36%	10	30	0	0	68,99%	18	20	2	0	6,19%	15	0	25	0	0,00%
	8	0	40	0	0	89,62%	0	40	0	0	89,19%	8	32	0	0	73,32%	19	21	0	0	7,10%	12	0	28	0	0,00%
	10	0	40	0	0	94,30%	0	40	0	0	94,99%	1	39	0	0	84,30%	14	25	1	0	7,29%	13	0	27	0	0,00%
	12	0	40	0	0	96,86%	0	40	0	0	96,20%	1	37	1	1	89,49%	11	28	1	0	9,86%	20	1	19	0	0,00%
	14	0	40	0	0	98,01%	0	40	0	0	97,67%	0	39	0	1	92,19%	9	29	2	0	10,31%	20	2	18	0	0,01%

CS		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		50	0	0	0	0,00%	50	0	0	0	0,00%	49	0	1	0	0,00%	45	0	5	0	0,00%	11	0	39	0	0,00%
2		4	46	0	0	51,53%	4	46	0	0	51,73%	2	48	0	0	54,20%	10	38	1	1	47,94%	11	5	34	0	4,95%
5		0	32	0	18	84,78%	0	28	0	22	83,26%	0	14	0	36	84,39%	0	18	0	32	80,73%	0	12	26	12	28,32%
10		0	30	0	20	93,30%	0	29	0	21	93,82%	0	23	0	27	94,99%	0	17	0	33	87,40%	0	24	16	10	42,57%
	6	14	26	0	0	44,98%	14	26	0	0	44,66%	12	28	0	0	47,21%	18	16	2	4	38,17%	4	11	25	0	27,48%
	8	10	30	0	0	55,78%	10	30	0	0	55,97%	10	19	0	11	45,73%	11	15	0	14	36,05%	4	7	27	2	23,52%
	10	10	27	0	3	58,60%	10	23	0	7	54,56%	10	17	0	13	47,07%	9	16	1	14	43,54%	4	6	27	3	19,97%
	12	10	13	0	17	40,97%	10	13	0	17	41,33%	9	11	1	19	39,73%	9	14	1	16	44,07%	5	7	19	9	18,44%
	14	10	12	0	18	43,05%	10	11	0	19	41,05%	10	10	0	20	39,82%	8	12	2	18	45,14%	5	10	17	8	23,26%

CP		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		50	0	0	0	0,00%	50	0	0	0	0,00%	49	0	1	0	0,00%	45	0	5	0	0,00%	11	0	39	0	0,00%
2		12	38	0	0	76,00%	12	38	0	0	76,00%	16	34	0	0	68,00%	27	22	1	0	37,25%	16	0	34	0	0,00%
5		0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	84,27%	1	22	27	0	35,68%
10		0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	85,49%	0	33	16	1	38,98%
	6	20	20	0	0	50,00%	20	20	0	0	50,00%	20	20	0	0	50,00%	18	20	2	0	45,08%	4	11	25	0	30,32%
	8	12	28	0	0	70,00%	12	28	0	0	70,00%	16	24	0	0	60,00%	20	20	0	0	43,10%	5	7	27	1	22,92%
	10	10	30	0	0	75,00%	10	30	0	0	75,00%	10	30	0	0	75,00%	15	24	1	0	51,72%	4	9	27	0	23,30%
	12	10	30	0	0	75,00%	10	30	0	0	75,00%	9	30	1	0	76,92%	11	28	1	0	60,54%	8	13	19	0	23,06%
	14	10	30	0	0	75,00%	10	30	0	0	75,00%	10	30	0	0	75,00%	8	30	2	0	65,86%	7	15	18	0	26,81%

Table 24: Correlated results, time widows with doors and trucks per door subsets

TI		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		9	41	0	0	57,43%	14	36	0	0	52,17%	23	27	0	0	31,04%	41	0	9	0	0,00%	10	0	40	0	0,00%
2		0	50	0	0	97,60%	0	50	0	0	97,34%	0	50	0	0	89,36%	28	20	2	0	5,95%	17	0	33	0	0,00%
5		0	50	0	0	99,98%	0	50	0	0	100,00%	0	50	0	0	97,91%	0	50	0	0	15,45%	25	1	24	0	0,01%
10		0	49	0	1	100,00%	0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	18,28%	36	3	11	0	0,04%
	6	8	32	0	0	73,32%	10	30	0	0	73,03%	10	30	0	0	68,10%	16	20	4	0	10,67%	16	0	24	0	0,00%
	8	1	39	0	0	84,17%	3	37	0	0	80,35%	9	31	0	0	72,71%	15	22	3	0	9,54%	14	1	25	0	0,06%
	10	0	40	0	0	93,30%	1	39	0	0	91,09%	4	36	0	0	79,06%	14	24	2	0	9,68%	14	0	26	0	0,00%
	12	0	40	0	0	94,98%	0	40	0	0	94,64%	0	40	0	0	85,27%	12	27	1	0	11,19%	24	0	16	0	0,00%
	14	0	39	0	1	97,94%	0	40	0	0	97,76%	0	40	0	0	92,75%	12	27	1	0	11,04%	20	3	17	0	0,03%

CS		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		50	0	0	0	0,00%	50	0	0	0	0,00%	50	0	0	0	0,00%	41	0	9	0	0,00%	10	0	40	0	0,00%
2		8	42	0	0	44,17%	8	42	0	0	42,19%	10	40	0	0	43,08%	12	36	2	0	34,88%	13	4	33	0	2,95%
5		0	32	0	18	82,86%	0	30	0	20	82,01%	0	16	0	34	82,75%	0	29	0	21	78,86%	1	18	24	7	21,35%
10		0	32	0	18	93,43%	0	46	0	4	95,73%	0	32	0	18	94,72%	0	22	0	28	86,12%	0	29	8	13	36,78%
	6	18	22	0	0	40,51%	18	22	0	0	41,12%	20	20	0	0	41,97%	16	19	4	1	41,44%	4	11	24	1	21,68%
	8	10	30	0	0	52,18%	10	29	0	1	51,03%	10	24	0	6	47,70%	11	16	3	10	37,86%	3	11	25	1	20,41%
	10	10	28	0	2	57,44%	10	28	0	2	56,49%	10	19	0	11	46,53%	8	21	2	9	41,84%	5	8	23	4	18,79%
	12	10	16	0	14	46,91%	10	22	0	8	55,38%	10	13	0	17	39,86%	9	15	1	15	41,45%	6	12	16	6	19,73%
	14	10	10	0	20	37,00%	10	17	0	13	52,57%	10	12	0	18	42,84%	9	16	1	14	47,01%	6	9	17	8	20,79%

CP		TW = 1.00					TW = 0.75					TW = 0.50					TW = 0.25					TW = 0.00				
n	$\frac{k}{n}$	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap	Opt.	Feas.	Infeas.	Unkn.	Gap
1		50	0	0	0	0,00%	50	0	0	0	0,00%	50	0	0	0	0,00%	41	0	9	0	0,00%	10	0	40	0	0,32%
2		13	37	0	0	74,00%	15	35	0	0	70,00%	19	31	0	0	62,00%	32	16	2	0	27,26%	17	0	33	0	0,01%
5		0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	81,54%	4	22	24	0	23,73%
10		0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	100,00%	0	50	0	0	82,53%	0	39	9	2	30,94%
	6	20	20	0	0	50,00%	20	20	0	0	50,00%	20	20	0	0	50,00%	16	20	4	0	46,05%	6	10	24	0	21,04%
	8	13	27	0	0	67,50%	15	25	0	0	62,50%	19	21	0	0	52,50%	17	20	3	0	45,02%	4	11	25	0	19,89%
	10	10	30	0	0	75,00%	10	30	0	0	75,00%	10	30	0	0	75,00%	17	21	2	0	44,17%	5	9	24	2	16,69%
	12	10	30	0	0	75,00%	10	30	0	0	75,00%	10	30	0	0	75,00%	12	27	1	0	57,20%	9	15	16	0	19,23%
	14	10	30	0	0	75,00%	10	30	0	0	75,00%	10	30	0	0	75,00%	11	28	1	0	58,44%	7	16	17	0	21,60%

Table 25: Unrelated results, time widows with doors and trucks per door subsets

is the worst. Finally, it appears that CP is poorly influenced by the number of trucks per door. The best results are obtained for instances with only 6 trucks per door. Worse but similar results are obtained for higher values. To conclude, CP is particularly efficient when there are only few doors and when time windows are tight.

UNRELATED RESULTS As before, a look at the results for unrelated instances does not show any different behaviour. Values slightly differ, but are similar on many points. The whole analysis realised on correlated instances also applies on unrelated ones.

9.2 SECOND TESTBED: QUADRATIC COMPATIBLE

Tables 26 and 27 present results for instances of the second testbed, with one door, and without TW (i.e. *1.00*). The column of relative gap does not appear anymore as all methods have solutions for almost all instances. This testbed shows the dominance of the quadratic method. QU behaves perfectly, with all instances optimally solved. Only the computation time attests of the increasing difficulty of the subsets. Because of the performances of QU, all optimal solutions of this testbed are known, and one can evaluate the solution (UB) of all methods relatively to the optimal solution. This difference from the optimal solution is reported in column "$\mathrm{UB}_{\mathtt{diff}}$ *OPT*".

As previously, correlated instances are analysed first.

TI, strong on TW *0.00*, has difficulties in solving the instances with 8 trucks per door or more. The solution found is not bad, from 0% to 13% from the optimal solution, but the LB is very bad, leading to a 100% gap. CS tends to get stuck at 18 trucks per door. From this point, it struggles to return solutions, up to 31% from the optimal solution with 48 trucks per doors. Finally we observe that CP encounters troubles at the same threshold, 18 trucks per door. However, as shows the $\mathrm{UB}_{\mathtt{diff}}$ OPT, the difficulty lies in finding good LB. Reaching 24 trucks per door, CP is not able to find all optimal solutions (without proving the optimality), but remains very close until the end of the testbed (-1% to -3% from the optimal).

UNRELATED RESULTS Expectingly, unrelated instances keep to behave as correlated ones. Values slightly differ, and are similar on many points. The whole analysis realised on correlated instances applies also on unrelated ones.

9.3 CONCLUSIONS

In the first part of this thesis we have introduced various methods for truck scheduling at an n-door crossdocking terminal, in which the objective is to minimize the total time spent in the system by the items

			TI							CS						
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time
1	6	1.00	9	1	0	0	2%	0%	4108,89	10	0	0	0	0%	0%	0,02
1	8	1.00	0	10	0	0	63%	0%	7220,08	10	0	0	0	0%	0%	0,07
1	10	1.00	0	10	0	0	79%	0%	7228,67	10	0	0	0	0%	0%	0,62
1	12	1.00	0	10	0	0	88%	-1%	7217,09	10	0	0	0	0%	0%	3,98
1	14	1.00	0	10	0	0	92%	-1%	7208,55	10	0	0	0	0%	0%	39,97
1	16	1.00	0	10	0	0	96%	-2%	7218,88	10	0	0	0	0%	0%	717,70
1	18	1.00	0	10	0	0	98%	-3%	7218,04	9	1	0	0	2%	0%	2338,99
1	20	1.00	0	10	0	0	98%	-3%	7220,89	4	6	0	0	12%	0%	5453,71
1	22	1.00	0	10	0	0	99%	-2%	7206,43	1	9	0	0	28%	-1%	6733,81
1	24	1.00	0	10	0	0	99%	-4%	7206,54	0	10	0	0	42%	-3%	7201,93
1	28	1.00	0	10	0	0	100%	-9%	7213,35	0	10	0	0	56%	-9%	7200,23
1	32	1.00	0	10	0	0	100%	-7%	7217,01	0	10	0	0	65%	-17%	7200,06
1	36	1.00	0	10	0	0	100%	-10%	7223,26	0	10	0	0	69%	-21%	7200,54
1	40	1.00	0	10	0	0	100%	-12%	7217,00	0	10	0	0	72%	-22%	7200,03
1	44	1.00	0	10	0	0	100%	-13%	7209,50	0	10	0	0	76%	-25%	7200,05
1	48	1.00	0	10	0	0	100%	-13%	7210,33	0	8	0	2	78%	-31%	7201,83
1	all	1.00	9	151	0	0	88%	-5%	7021,53	74	84	0	2	31%	-8%	4105,85
			CP							QU						
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time
1	6	1.00	10	0	0	0	0%	0%	0,32	10	0	0	0	0%	0%	0,00
1	8	1.00	10	0	0	0	0%	0%	1,24	10	0	0	0	0%	0%	0,00
1	10	1.00	10	0	0	0	0%	0%	7,52	10	0	0	0	0%	0%	0,01
1	12	1.00	10	0	0	0	0%	0%	27,55	10	0	0	0	0%	0%	0,01
1	14	1.00	10	0	0	0	0%	0%	290,10	10	0	0	0	0%	0%	0,02
1	16	1.00	10	0	0	0	0%	0%	1476,17	10	0	0	0	0%	0%	0,05
1	18	1.00	5	5	0	0	36%	0%	6297,21	10	0	0	0	0%	0%	0,13
1	20	1.00	0	10	0	0	73%	0%	7200,00	10	0	0	0	0%	0%	0,71
1	22	1.00	0	10	0	0	77%	0%	7200,00	10	0	0	0	0%	0%	0,75
1	24	1.00	0	10	0	0	78%	-1%	7200,00	10	0	0	0	0%	0%	2,12
1	28	1.00	0	10	0	0	79%	-2%	7200,00	10	0	0	0	0%	0%	5,96
1	32	1.00	0	10	0	0	80%	-2%	7200,00	10	0	0	0	0%	0%	14,38
1	36	1.00	0	10	0	0	80%	-2%	7200,00	10	0	0	0	0%	0%	28,83
1	40	1.00	0	10	0	0	82%	-2%	7200,00	10	0	0	0	0%	0%	43,03
1	44	1.00	0	10	0	0	82%	-3%	7200,00	10	0	0	0	0%	0%	188,03
1	48	1.00	0	10	0	0	82%	-1%	7200,00	10	0	0	0	0%	0%	357,96
1	all	1.00	65	95	0	0	47%	-1%	4556,26	160	0	0	0	0%	0%	40,13

Table 26: Correlated results, focus on the quadratic subset

			TI							CS						
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time
1	6	1.00	8	2	0	0	2%	0%	3515,13	10	0	0	0	0%	0%	0,03
1	8	1.00	1	9	0	0	39%	0%	7200,86	10	0	0	0	0%	0%	0,06
1	10	1.00	0	10	0	0	74%	0%	7214,30	10	0	0	0	0%	0%	0,27
1	12	1.00	0	10	0	0	80%	0%	7210,44	10	0	0	0	0%	0%	1,12
1	14	1.00	0	10	0	0	92%	-1%	7213,43	10	0	0	0	0%	0%	8,56
1	16	1.00	0	10	0	0	95%	-3%	7215,59	10	0	0	0	0%	0%	125,15
1	18	1.00	0	9	0	0	98%	-1%	7217,35	10	0	0	0	0%	0%	659,03
1	20	1.00	0	10	0	0	97%	-4%	7215,25	4	6	0	0	6%	0%	4542,47
1	22	1.00	0	10	0	0	98%	-3%	7218,73	3	7	0	0	15%	0%	5524,29
1	24	1.00	0	10	0	0	99%	-7%	7210,49	0	10	0	0	35%	-2%	7203,66
1	28	1.00	0	10	0	0	100%	-6%	7212,54	0	10	0	0	48%	-7%	7204,39
1	32	1.00	0	10	0	0	100%	-11%	7212,87	0	10	0	0	61%	-15%	7200,18
1	36	1.00	0	10	0	0	100%	-12%	7208,44	0	10	0	0	65%	-13%	7200,09
1	40	1.00	0	10	0	0	100%	-10%	7213,81	0	10	0	0	70%	-20%	7200,37
1	44	1.00	0	10	0	0	100%	-16%	7207,10	0	10	0	0	75%	-23%	7200,10
1	48	1.00	0	10	0	0	100%	-14%	7219,11	0	9	0	1	77%	-29%	7200,67
1	all	1.00	9	150	0	0	86%	-6%	6980,11	77	82	0	1	28%	-7%	3829,40
			CP							QU						
n	$\frac{k}{n}$	TW	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time	Opt.	Feas.	Infeas.	Unkn.	Gap	UB_{diff} OPT	Time
1	6	1.00	10	0	0	0	0%	0%	0,14	10	0	0	0	0%	0%	0,01
1	8	1.00	10	0	0	0	0%	0%	0,48	10	0	0	0	0%	0%	0,01
1	10	1.00	10	0	0	0	0%	0%	2,67	10	0	0	0	0%	0%	0,01
1	12	1.00	10	0	0	0	0%	0%	13,73	10	0	0	0	0%	0%	0,01
1	14	1.00	10	0	0	0	0%	0%	108,39	10	0	0	0	0%	0%	0,02
1	16	1.00	10	0	0	0	0%	0%	411,58	10	0	0	0	0%	0%	0,04
1	18	1.00	10	0	0	0	0%	0%	3149,82	10	0	0	0	0%	0%	0,13
1	20	1.00	3	7	0	0	48%	0%	6212,78	10	0	0	0	0%	0%	0,72
1	22	1.00	0	10	0	0	74%	0%	7200,00	10	0	0	0	0%	0%	0,76
1	24	1.00	0	10	0	0	77%	-1%	7200,00	10	0	0	0	0%	0%	2,13
1	28	1.00	0	10	0	0	76%	-2%	7200,00	10	0	0	0	0%	0%	5,98
1	32	1.00	0	10	0	0	79%	-5%	7200,00	10	0	0	0	0%	0%	14,40
1	36	1.00	0	10	0	0	80%	-1%	7200,00	10	0	0	0	0%	0%	28,79
1	40	1.00	0	10	0	0	81%	-7%	7200,00	10	0	0	0	0%	0%	42,86
1	44	1.00	0	10	0	0	83%	-3%	7200,00	10	0	0	0	0%	0%	187,98
1	48	1.00	0	10	0	0	82%	-3%	7200,00	10	0	0	0	0%	0%	358,64
1	all	1.00	73	87	0	0	42%	-1%	4218,72	160	0	0	0	0%	0%	40,15

Table 27: Unrelated results, focus on the quadratic subset

being transfered, and there exist both precedence constraints among trucks and time window constraints. The problem appears computationally challenging, as even with a single door the problem is NP-hard, for general truck loading/unloading times. We have developed four different formulations (time-indexed, critical set, constraint programming and integer quadratic programming) and performed extensive computational experiments to assess their performance.

It turns out that considering the quality of the solution, CP works better on all instances. This formulation is particularly efficient when time windows are not binding or when the number of doors is low. The weakness of CP lies in the bad quality of its LB. The TI method has higher UB than CP, but also higher LB, especially for very tight time windows. The CS formulation finds the best LB but is relatively poorly suited to find good solutions.

The distinction between correlated and uncorrelated instances has no impact on the calculations. Unexpectedly, the complexity introduced by independent load and loading time is not observable for any used methods.

Part III

ROBUSTNESS AND GROUPS OF PERMUTABLE TASKS

10

STATE OF THE ART - CROSSDOCK TRUCK SCHEDULING UNDER UNCERTAINTIES

10.1 UNCERTAINTIES IN SCHEDULING

The CTSP being a scheduling problem in essence, this section sums up the main concepts related to scheduling under uncertainties. Scheduling concerns a large variety of problems, and sources of uncertainties are numerous. We refer to papers or books dedicated to robustness in scheduling which present the various uncertainty sources, as well as methods to handle them. For instance, Billaut et al. (2008), themselves referring to the book of Kouvelis and Yu (1997), mention the following sources of uncertainties: tasks duration, release and due dates, machine failures and production costs. Clearly, not all of them have to be addressed at the same time as their importance and frequency depend on the context. Relevant uncertainties can be a priori taken into account in the problem and there exist different methods to tackle them. The most popular ones model them as stochastic variables, intervals, discrete scenarios, or a mix of these representations.

To solve a problem with uncertainties, different approaches are used. We distinguish between:

- Proactive methods. The schedule is built in the offline phase, trying to determine a *robust schedule* that minimizes the impact of the uncertainties. This can be achieved by considering the most probable scenario, computing a schedule remaining efficient for any scenario, and/or adding idle time in the schedule to absorb the changes that will occur in the online phase.

- Reactive methods. Assuming that uncertainties are hardly predictable, the schedule is built online, when events appear. These schedules are often realized with priority rules, like first-in/first-out, trying to ensure a performance guarantee.

- Proactive-reactive methods. A first robust schedule, or a set of schedules is computed in the offline phase taking explicitly the uncertainties into account. This schedule is dynamically updated during the online phase, taking the realized knowledge into account. There are several ways to adapt a robust schedule that depends on the structure. For instance, one can simply use rules to take benefit of time buffers that have been inserted in the schedule proactively. If several schedules are determined in the offline phase, one can also select the best one taking into account the real data. Finally, one can try to reactively repair the

robust schedule in order to find the closest schedule, that is still feasible with respect to the actual situation.

Figure 24 illustrates the application of the various methods.

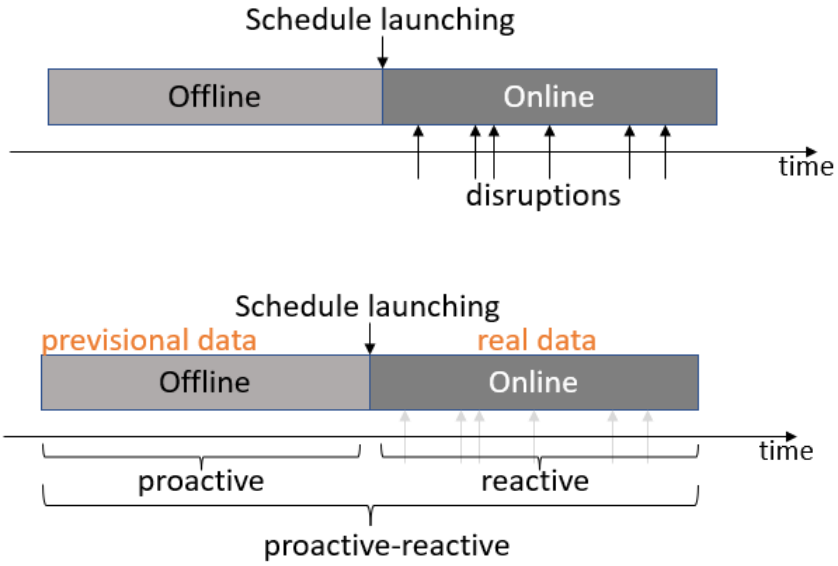

Figure 24: Proactive and reactive method

Many proactive methods insert flexibility into the schedule to face unforeseen events so that the schedule of the tasks can be dynamically adapted, by example tasks can be delayed. Four types or flexibility are defined.

- Time flexibility allows tasks to be moved respecting their initial order;
- Sequence flexibility allows to modify the sequence of the tasks on each resource (which implies time flexibility);
- Resource flexibility allows to modify the ssignment of the tasks to the resources;
- Model flexibility allows to modify some parameters of the model, like preemption, or deadlines.

A method/solution is said robust if it is able to undergo uncertainties, possibly using flexibility, keeping the performance degradation under control. However, the way robustness or flexibility are measured has to be defined for every problem as these metrics intrinsically change the nature of the approach.The reader is advised to consult the book of Billaut et al. (2008) for further information about measures of flexibility and robustness.

10.2 CROSSDOCK TRUCK SCHEDULING UNDER UNCERTAINTIES

The deterministic version of the CTSP having been addressed in chapter 2, this section focuses on its non-deterministic version.

As mentioned in the beginning of this chapter, sources of uncertainty are numerous and quite specific to the considered problem. In the literature, the articles related to robust crossdock scheduling address different kinds of uncertainties. Nevertheless, most of them concentrate on truck arrival times which are the most impacting contingencies encountered in the crossdock management. As pointed-out by Boysen and Fliedner (2010) (a relevant and frequently cited paper of the crossdocking literature): "Arrival times of trucks are typically bound to heavy inaccuracies, because traffic congestion or engine failures delay inbound trucks". Thus uncertainties on arrival times of trucks is tackled in (Larbi et al., 2011; Heidari et al., 2018; Rajabi and Shirazi, 2016; Zouhaier et al., 2016). In (Konur and Golias, 2013b) and (Konur and Golias, 2013a), the authors consider uncertain arrival times for inbound trucks only.

Other authors consider both uncertainties on arrival times and on processing times, e.g. Xi et al. (2020) and Ladier and Alpan (2016b). Amini and Tavakkoli-Moghaddam (2016) consider truck breakdowns as the main failure source, which induces arrival lateness.

The next chapters will also consider the truck arrival times as the main source of uncertainty.

To handle uncertainties, a large variety of methods has been used through the literature. A large part of them are purely proactive methods: a schedule is built before the execution of the planning, with a lack of knowledge on the data. Konur and Golias (2013a) and Konur and Golias (2013b) propose a set of approaches to solve the problem when considering a set of possible values for the arrival dates of inbound trucks. Their objective is to use a cost-stable scheduling strategy by minimizing the average of total service costs. A bi-objective bi-level optimization problem is formulated and a genetic algorithm based heuristic to find Pareto efficient schedules is proposed. Similarly, Heidari et al. (2018) consider a model with uncertainties on the arrival times of inbound and outbound trucks. They also aim at finding a cost-stable scheduling strategy ans also propose metaheuristics in the spirit of the work of Konur and Golias, 2013b. Rajabi and Shirazi (2016) also consider uncertainties on arrival times and aim to minimize the waiting time. The authors propose chance-constrained mathematical programming to solve the problem.

Ladier and Alpan (2016b) consider both uncertainties on processing times and arrival times. Their solution method relies on a min-max method, minimizing the expected regret as well as resource and time redundancies. Xi et al. (2020) also consider both uncertainties on processing times and arrival times. They aim to minimize the total

cost and the number of conflicts, corresponding to a potential overlap of trucks in the worst case. Amini and Tavakkoli-Moghaddam (2016) consider truck breakdowns as uncertainty sources. The objective is to minimize both the total tardiness and the total weighted completion time. A stochastic model is developed; the probability of breakdown follows a Poisson distribution. Meta-heuristics are proposed to solve the model.

To the best of our knowledge, only Larbi et al. (2011) take interest in a reactive method for the CTSP that differs according to the level of knowledge on the arrival dates of the trucks. Thus, without any prior information on the arrival dates, the scheduling is realized online, according to priority rules. Heuristics are proposed to solve the problem in a restricted case having only a single inbound and a single outbound door.

Finally, a proactive-reactive method is proposed by Zouhaier et al. (2016). Dealing also with uncertainties on arrival times, the authors develop the so-called truck appointment system, a method that minimizes the number of incoming trucks per slot and balances the workload between platforms according to the evolution of the demand. Truck companies can book time slots and participate to the initialization of a schedule. The truck appointment system estimates the workload of doors and adjusts truck appointment times according to the actual arrival dates. In this way, the schedule is dynamically adapted according to event occurrences. The objective is to minimize the time gap between the realized handling time of a truck and its original appointment, taking also the waiting cost of the trucks into account. According to the authors, this proactive-reactive method performs very well in practice.

The method proposed in this work to face arrival time uncertainties i.s of proactive-reactive nature. It uses the concept of a group of permutable tasks. To the best of our knowledge, this is the first attempt to apply groups of permutable tasks to the CTSP. The following subsection focuses on the literature related to scheduling with groups of permutable tasks.

10.3 GROUPS OF PERMUTABLE TASKS

As the offline and online methods can be designed in a complementary way, proactive-reactive methods can bring interesting results in practice, while they are not often addressed in the literature, compared to purely proactive or reactive ones. Van de Vonder et al. (2007) underlines the efficiency of the proactive-reactive compared to the proactive method with a comparison study. Proactive-reactive methods using groups of permutable tasks have been studied for years. Let us first mention the work of Le Gall (1989) who applied this concept on workshop problems and proved its interest, as still testified

by some recent publications on this subject. In a schedule with groups of permutable tasks, a sequence of groups is determined for each resource, provided that the order of the tasks in a group is not fixed. This partial schedule can be used reactively as the order of the tasks in a group can be chosen opportunistically by a reactive algorithm, according to the actual event occurrences. In the proactive phase, the basic objective is to make a good compromise between flexibility and performance, i.e. one looks for a group sequence that balances the number of groups with the worst sojourn time. Flexibility is indeed defined by the number of groups, because fewer groups imply a larger cardinality, thus more permutations. Wu et al. (1999) consider an "Ordered Assignment and Detailed Scheduling Problems", where they use the same idea of groups of permutable tasks in order to add robustness to a job shop solution. Ashtiani et al. (2011), considering stochastic task durations, use a "pre-processing" phase on the RCPSP which makes a number of a-priori sequencing decisions while the remaining decisions are taken dynamically during project execution.

The worst sojourn time computation is pessimistic as it consists in finding the worst sequence being characterized, assuming that such a sequence can be reached online by the reactive algorithm. In e.g. (Billaut et al., 2008) and (Artigues et al., 2016), extensions of the concept of groups of permutable tasks are proposed evaluate the worst sequence with respect to some specific reactive algorithms.

Billaut et al. (2008) introduce two methods: ORABAID, developped years ago, and AMORFE, a more recent one. These methods specifically consider the jobshop scheduling environment where one aims at minimizing the maximum lateness, but they can be adapted to other scheduling problems. ORABAID is a heuristic method, with a proactive phase decomposed in two steps: i) the computation of a group sequence thanks to a heuristic and ii) the improvement of that sequence to find a better flexibility/performance compromise. The ORABAID method measures the quality of the group sequence based on the worst lateness case. In the AMORFE method, the proactive phase is updated in order to take both the worst and best latenesses into account in the design of the group sequence. While the worst case gives a performance guarantee, the authors highlight that the best case is also of importance in the estimation of the quality of the group sequence. Indeed, as it is very unlikely that the worst case will happen, working with intervals of expected performances could be better suited. Artigues et al. (2016), modeling uncertainties via a discrete set of scenarios, realize a review of the models and algorithms that have been proposed in the literature for evaluating a group sequence. Examples based on single machine and jobshop scheduling are studied, where the quality of the group sequence is determined by the Best Earliest Schedule, the Best Latest Schedule, the Worst Earliest Schedule, and the Worst Latest Schedule with respect to a given

set of scenarios. These four sub-problems are defined in (Artigues et al., 2016) for a regular time objective. A MILP formulation and a heuristic method for the maximum lateness minimization on the single machine problem are also presented. Yahouni (2019) focuses on the best case evaluation in his thesis. The considered objective is the minimization of the makespan and, even with a regular time objective, the problem is NP-hard. A heuristic called EBJG is proposed for the design of group sequences. This work follows the work of Pinot and Mebarki (2009), who propose exact methods to solve the best case problem, which can be assimilated to a longest path calculation in a jobshop. Neufeld et al. (2016) complete this framework with a review of the literature on groups applied on flowshop scheduling.

In the previously cited papers, the studied objectives are regular and time oriented (lateness, makespan, etc.). As already mentioned in Chapter 2, the sojourn time objective being relevant in the crossdock environment, the next sections are devoted to its study in the scheduling context of CTSP with groups sequences.

11

USING GROUPS OF PERMUTABLE TASKS FOR THE CTSP UNDER UNCERTAINTY

11.1 PROBLEM DESCRIPTION

The studied problem is a variation of the CTSP defined in section 3.1 where uncertainties on release dates are now considered. The reader can refer to this section for the definition of the notations used for the problem that are slightly adapted in this section. The variations are detailed below. Also, as we did not observe previously any particular differences between the complexity of solving correlated versus unrelated CTSP versions, we only focus from now on the correlated version.

For each truck u, we define a time interval $\Omega_u = [\underline{r_u}; \overline{r_u}]$ including the expected release date r_u. This interval basically models the uncertainty related to the arrival time as an equiprobable interval of values. Indeed, from the real life practices, modelling the uncertainty of arrivals is very important as it is a common source of encountered disruptions, and some disruption can ruin the planned schedule. We further refer to $\tilde{r}_u$ as the realized value of the arrival time of truck u. (In the deterministic CTSP, $r_u = \tilde{r}_u$.)

The sojourn time objective is considered as the main objective. Nevertheless, as elaborated later, the minimization of the lateness L_{max} is also used as a secondary objective. This means that the deadlines of the initial CTSP are now relaxed and seen as due dates. The lateness of a truck u corresponds to the amount of time by which the due date is violated $L_u = \tilde{r}_u + p_u - d_u$. Clearly, these objectives are conflicting as it can be suitable to delay the unloading of an inbound truck, hence to increase its lateness, in order to improve the sojourn time.

11.2 USING GROUPS OF PERMUTABLE TASKS

The decision to use the concept of groups of permutable tasks has been taken to solve our uncertain CTSP. We will devise a proactive-reactive method that embraces both the offline and online phases, as illustrated in Figure 25.

During the offline phase, a flexible schedule is determined, fixing a partial order on each door. Afterwards, in the online phase (during the implementation of the planning), the final schedule is dynamically consolidated, accordingly to the real early and late arrivals of trucks, provided that the final total order of the trucks is an exten-

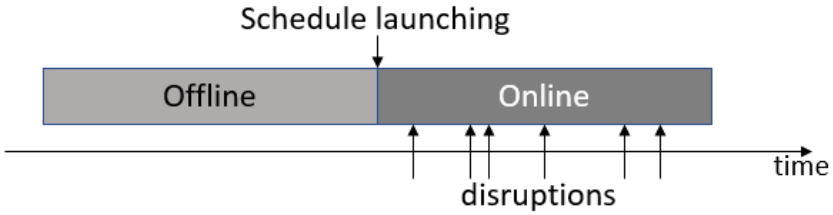

Figure 25: Illustration of the online and offline mode

sion of the partial order defined in the proactive phase. To create a flexible schedule, groups of permutable tasks are used. A group of permutable tasks contains a set of trucks that are allocated to the same door and are sequenced before or/and after other groups on the same door. Figure 26 shows an example of a group containing three trucks that can be sequenced in any order. Let us highlight that the realization of a release date for a truck u may possibly delay some trucks sequenced after u, according to the chosen sequence inside the group, as illustrated in the previous figure. The offline phase aims to set up a group sequence $\mathcal{G}$, as illustrated in Figure 27. Obviously, as an inbound truck i sharing a handover relation with an outbound truck o cannot be sequenced after o on the same door, they cannot be assigned to the same group.

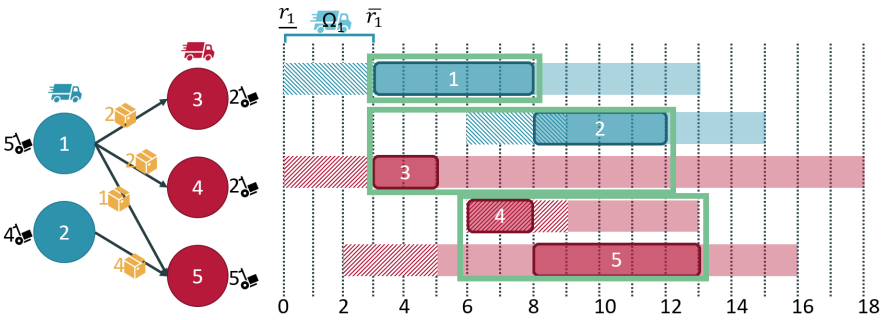

Figure 26: Group example

Figure 28 illustrates the online phase, with the execution of the planning, and the progressive consolidation of the sequence of trucks chosen inside a group as they arrived. Whenever a truck belonging to a group g enters the crossdock, one might start its handling while its door d is available, provided that all trucks belonging to the group immediately preceding g have all been completed on d. Figure 29 illustrates the total sequence and the corresponding schedule after the implementation of the planning. This resulting schedule brings the value of the objectives.

Let us introduce a few notations. We further refer to $\mathcal{G}$ as a specific set of n group sequences of permutable trucks, to g as a specific

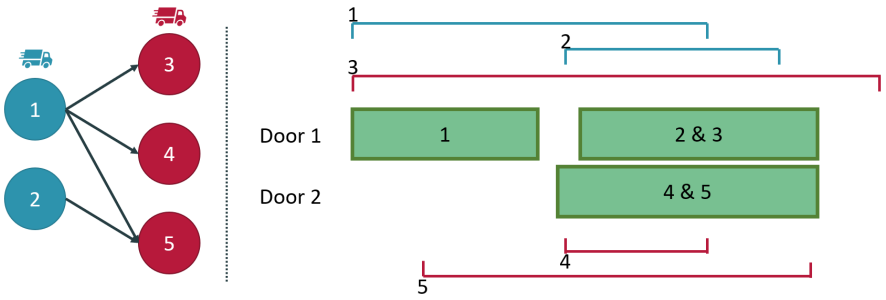

Figure 27: Group sequence

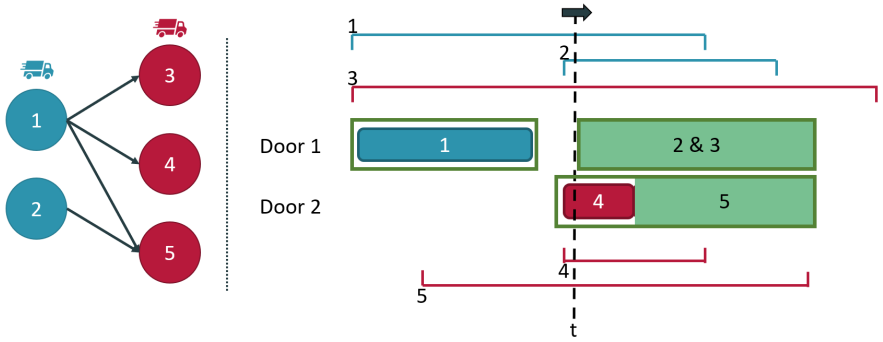

Figure 28: Consolidation of the total order during the online phase

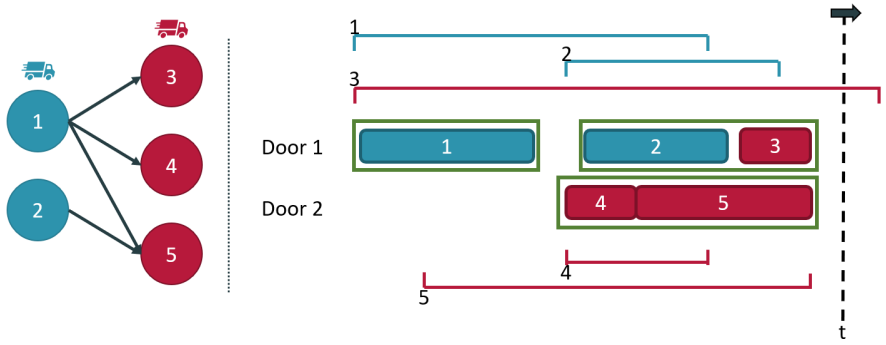

Figure 29: Resulting schedule

group $\in \mathcal{G}$, to $g(u)$ as the particular group $\in \mathcal{G}$ containing truck u, to $g^-(u)$ as the group preceding $g(u)$ on the same door, and $g^+(u)$ as the group following $g(u)$.

In order to illustrate the need of the second L_{max} objective and to have a better understanding of the way of using groups, let us have a look at the same example as previously with 2 inbound trucks (blue trucks 1 and 2), 3 outbound trucks (red trucks 3, 4 and 5), and 2 doors. The handover constraints between trucks are presented at the left side of Figure 29. The right side of this figure indicates:

- the uncertainty interval of each truck, which is displayed with colored lines at the top or the bottom of the tasks;

- the door-to-truck assignment: Trucks 1, 2, and 3 are assigned to door 1, while trucks 4 and 5 are on door 2;
- the structure of the group sequence: Truck 1 forms a group on its own, while trucks 2 and 3 form a second trucks group, and trucks 4 and 5 form a third group;

This group sequence characterizes 4 sequences in total, with two possibles permutation of trucks 2-3 and trucks 4-5. An earliest start schedule with $2 \prec 3$ and $4 \prec 5$ is represented in Figure 29, assuming the realization of the optimistic release dates. Figure 30 represents an alternative earliest start schedule with $3 \prec 2$ and $5 \prec 4$. Clearly, this latter schedule violates the due date of truck 4, which is now late, while the former respect all the due dates. This small example illustrates the interest of relaxing the initial deadlines, transforming them into due dates, to better take advantage of the group concept, without worrying about the feasibility of the schedule.

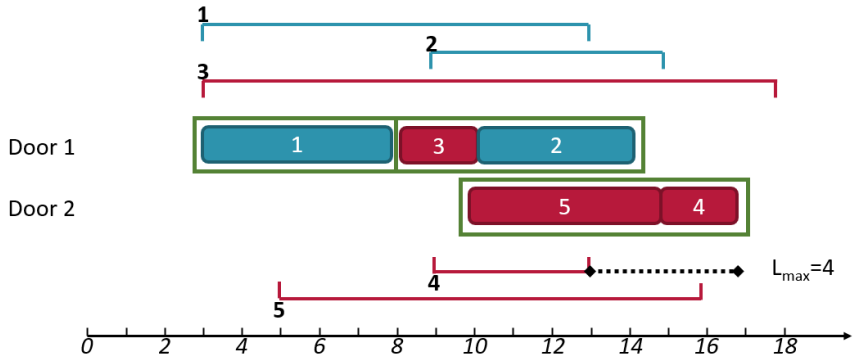

Figure 30: Small detailed example: Optimal maximum lateness

In the next chapter, we will use the following notations:

- ψ refers to the sojourn time;
- ψ^* denotes an optimal sojourn time, for a given total order and given realized release dates, i.e. $\min_{s_i, s_o} \psi(s_i, s_o)$,
- L_{max} is the maximum lateness of a truck schedule.

12 EVALUATION OF A GROUP SEQUENCE

To efficiently create groups and sequence them on each door, the ability to measure and compare the quality of various group sequences, in terms of lateness and sojourn time, is required. A sequence of groups possibly characterizes a lot of total truck orders, each of them defining to a lot of schedules, depending on the realization of the release dates. It is not relevant to enumerate all of them for the computation of the objective values, as they are over-numerous. We will focus on the worst and best case values of the optimum lateness and sojourn time. These values provide an interval containing the objective values of any solution characterized by the group schedule. As previously mentioned, the worst case is commonly used in robustness, as it offers a performance guarantee: whatever the realized release dates respecting the uncertainty intervals, the objective value cannot be worse than its worst case value. The best case is a second interesting performance measure that reflects the most optimistic value that can be expected.

12.1 BEST CASE SCENARIOS TO ESTIMATE THE QUALITY OF A GROUP SEQUENCE

12.1.1 *Best sojourn time*

As defined before, we are focusing on the weighted sum of the sojourn times of the pallets stocked in the warehouse. This section addresses the problem of computing the best value of such an objective. Obviously, for this computation, only the release date scenario such that trucks arrive as soon as possible need to be considered. Indeed, considering another scenario with greater arrival times can only lead to an increase of the sojourn time because the former (optimistic) scenario is a relaxation of the latter.

The best sojourn time can be defined as follows, considering the optimistic release date scenario:

$$\min_{r} \min_{\theta} \min_{s_i, s_o} \sum_{o \in \mathcal{O}} W_o . s_o - \sum_{i \in \mathcal{I}} W_i . s_i \tag{38}$$

where $\min_{r}$ means that for each truck u, $r_u = \underline{r}_u$ and where θ is a total order compatible with the group sequence.

This problem can be viewed as a parallel machine scheduling problem with release dates $\underline{r}$, deadlines $\tilde{d}$, and precedence constraints,

which is a generalization of $1|\underline{r}, \tilde{d}|$. The latter being NP-complete (Lenstra et al., 1977), finding a feasible solution for the best sojourn time is also NP-complete.

As learned in the second part of this manuscript, constraint programming is efficient on the CTSP to minimize the sojourn time. Therefore, a CP formulation can be used to solve that problem. The best sojourn time problem can be formulated as follows. The decision variables are the start times s_u for (un)loading truck u. We introduce a dummy truck o so that s_0 defines the origin of the time horizon with a processing time $p_0 = 0$. U_0 refers to the set of initial trucks extended with truck 0.

$$\min \sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \tag{39}$$

subject to

$$s_v - s_u \geqslant l_{uv} = \begin{cases} p_u & \forall u \in \mathcal{U}, \forall v \in g^+(u) \\ 0 & \forall (u,v) \in \mathcal{A} \\ \underline{r_v} & \forall v \in \mathcal{U}, u = 0 \\ -\tilde{d}_u + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \tag{40}$$

$$|J_{t,G}| \leqslant 1 \qquad \forall G \in \mathcal{G}, \forall t \in [0;T] \tag{41}$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \tag{42}$$

where $J_{t,G} = \{j \in G \,|\, s_u < t \leqslant s_u + p_u\}$ is a set containing all trucks j of group G that are being (un)loaded at time t.

Constraints (40) impose end-start precedences between trucks of different groups that follow one another on a dock, start-start precedence between inbound and outbound trucks and time-windows. Constraints (41) ensure that only one truck per group is (un)loaded at the same time.

12.1.2 *Best maximum lateness* L_{max}

On time truck departure is also a very important element in cross-docking. In order to evaluate this criterion we will look at the maximum lateness L_{max}. Given a total order, L_{max} is defined as the biggest lateness over all trucks.

$$L_{max} = \max_{u \in \mathcal{U}} (s_u + p_u - d_u)$$

We will therefore evaluate the best L_{max} given a group sequence. Again, for dominance reasons, the most favourable scenario of release dates is considered, i.e. all trucks arrive as soon as possible.

The best maximum lateness, with the best total order and the best release dates is defined as follows:

$$\min_{r} \min_{\theta} \min_{s_i, s_o} L_{max} \tag{43}$$

This problem is NP-hard (Artigues et al., 2005).

As the best sojourn time problem, the best maximum lateness problem can be written with a Constraint Programming formulation as follows:

$$\min\ L_{max} \tag{44}$$

subject to

$$s_v - s_u \geqslant l_{uv} = \begin{cases} p_u & \forall u \in \mathcal{U},\ \forall v \in g^+(u) \\ 0 & \forall (u,v) \in \mathcal{A} \\ \underline{r_v} & \forall v \in \mathcal{U}, u = 0 \\ -L_{max} - d_u + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \tag{45}$$

$$|J_{t,G}| \leqslant 1 \qquad \forall G \in \mathcal{G}, \forall t \in [0;T] \tag{46}$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \tag{47}$$

$$L_{max} \in \mathbb{R} \tag{48}$$

where $J_{t,G} = \{u \in G \,|\, s_u < t \leqslant s_u + p_u\}$.

Constraints (45) are similar to (40), however, the deadline is relaxed here because tasks are allowed to finish L_{max} time units after the deadline. As before, constraints (46) ensure that only one truck per group is (un)loaded at the same time.

12.2 WORST CASE SCENARIOS TO ESTIMATE THE QUALITY OF A GROUP SEQUENCE

12.2.1 *Worst optimal sojourn time*

This section takes interest in finding the optimal sojourn time (which should be minimized) in the worst possible arrival scenario. Clearly, only the scenario in which the trucks arrive as late as possible need to be considered. Indeed, for such a scenario, reducing any of the release dates can only result in a decrease of the optimal sojourn time. Therefore, the search reduces to finding the worst total order with respect to the optimal sojourn time.

The worst sojourn time can be hence defined as follows:

$$\max_{r} \max_{\theta} \min_{s_i, s_o} \sum_{o \in \mathcal{O}} W_o . s_o - \sum_{i \in \mathcal{I}} W_i . s_i \tag{49}$$

where $\max_{r}$ means that for each truck u, $r_u = \overline{r_u}$.

Due to this non-regular time objective, finding the worst sojourn time given a group sequence is NP-complete:

Theorem 7. *Given a general crossdock problem with a group sequence $\mathcal{G}$, finding a total order θ that minimizes the sojourn time ψ is NP-Complete.*

Proof. We reduce the single machine scheduling problem $1|r_j|\sum C_j$, which is known to be NP-Hard, to our problem.

We consider a single machine scheduling problem were the operations are indexed $1,\dots,k$ and we define p^s_j as the processing time and r^s_j as the release date of the operation indexed j.

Let us construct a crossdock problem instance with 2 doors, k inbound trucks indexed i such that $p_i = p^s_i - 1$ and $r_i = r^s_i$, for $i = 1,\dots,k$ and k outbound trucks indexed o such that $p_o = p^s_o$ and $r_o = r^s_o$, for $o = 1,\dots,k$. We assume that there is an initial stock containing $K = \sum p_o - \sum p_i$ pallets. Moreover, we assume that inbound truck indexed i delivers its p_i pallets to outbound truck indexed $o = i$ (i.e., $w_{io} = p_i = p_o - 1$ if $o = i$ else $w_{io} = 0$) and that each outbound truck indexed o receives exactly 1 pallet from the initial stock. The group sequence $\mathcal{G} = \{G_1, G_2\}$ is such that the sequence for the first door contains one group in which we can find all inbound trucks, while the sequence for the second group contains one group in which we can find all outbound trucks. The sojourn time can be expressed as

$$\psi = \sum_o (s_o - t_0) + \sum_i \sum_o w_{io}(s_o - s_i). \tag{50}$$

The former term corresponds to the cost of the pallets transferred directly from the stock to the outbound trucks where t_0 defines the beginning of the schedule. The latter is the cost of the pallets moved from inbound to outbound trucks.

Let us observe that, for any feasible outbound schedule for the second door, it is always possible to build a similar inbound schedule on the first door by defining $s_i = s_o$ if $i = o$ (because $p_i < p_o$ and $r_i = r_o$). Therefore, ψ reduces to $\sum_o (s_o - t_0)$, which equals $\sum_o C_o - K$ where $K = \sum_o (p_o + t_0)$. Hence, minimizing ψ is equivalent to solving a $1|r_j|\sum C_j$ problem on the second door. □

From Theorem 7, it results that finding a total order that gives the best or the worst sojourn time is NP-Complete (as the sub-problem is itself NP-Complete).

A schedule with worst optimal sojourn time is illustrated Figure 31, considering the example introduced Figure **??**. Let us observe that delaying some outbound truck can further increase the worst sojourn time but, in such a case, it will not be optimum any more.

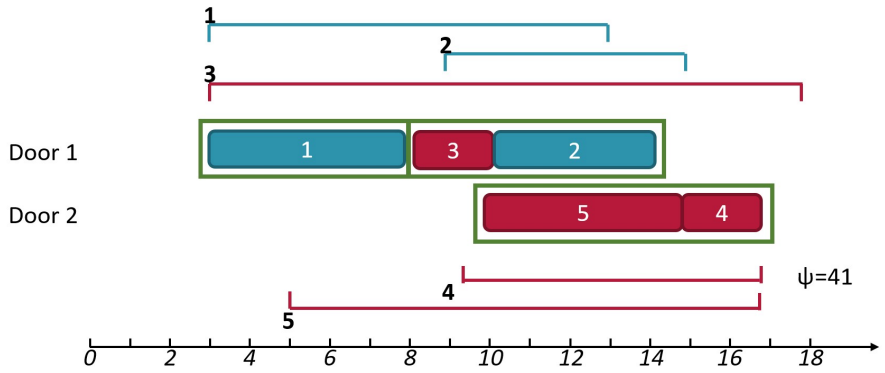

Figure 31: Worst sojourn time

Due to the max / min objective, finding the worst optimal sojourn time is more tricky. For that reason, the explanation is given in two steps.

12.2.1.1 *Finding the optimal sojourn time of totally ordered sequences*

First of all, we recall that $\max_r \psi^*(r) = \psi^*(\overline{r})$, i.e. the worst optimal sojourn time can be systematically obtained when the realized release dates are at the latest.

The well-known strong duality concept related to Linear Programming can be used to determine $\max_\theta \min_{s_i,s_o} \psi(s_i, s_o, \overline{r})$, as detailed below. We propose to use a primal-dual approach to solve the problem, as the objective is defined as a min-max.

As a first step, let us suppose that a total order has been chosen. Determining the optimal sojourn time knowing the total order of the trucks on the doors is a polynomial problem that can be efficiently solve with linear programming using the following (primal) formulation:

Primal problem

$$\min \sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \tag{51}$$

subject to

$$s_v - s_u \geqslant l_{uv} = \begin{cases} p_u & \forall u, v \in \mathcal{U}, u \prec v \\ 0 & \forall (u,v) \in \mathcal{A} \\ \overline{r_v} & \forall v \in \mathcal{U}, u = 0 \\ -\tilde{d_u} + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \tag{52}$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \tag{53}$$

Let us now consider the dual counterpart of the previous (primal) LP.

Dual problem

$$\max \sum_{\substack{(u,v)\in\mathcal{U}\\ u\prec v}} \phi_{uv}.p_u + \sum_{v\in\mathcal{U}} \phi_{0v}.\overline{r_v} + \sum_{u\in\mathcal{U}} \phi_{u0}.(-\tilde{d_u} + p_u) \tag{54}$$

subject to

$$\sum_{v\in\mathcal{U}} \phi_{v0} - \sum_{v\in\mathcal{U}} \phi_{0v} = 0 \tag{55}$$

$$\phi_{0i} - \phi_{i0} + \sum_{\substack{u\in\mathcal{U}\\ u<i}} \phi_{ui} - \sum_{\substack{v\in\mathcal{U}\\ i<v}} \phi_{iv} - \sum_{o\in\Gamma^+(i)} \phi_{io} = -W_i \qquad \forall i \in \mathcal{I} \tag{56}$$

$$\phi_{0o} - \phi_{o0} + \sum_{\substack{u\in\mathcal{U}\\ u<o}} \phi_{uo} + \sum_{i\in\Gamma^-(o)} \phi_{io} - \sum_{\substack{v\in\mathcal{U}\\ o<v}} \phi_{ov} = W_o \qquad \forall o \in \mathcal{O} \tag{57}$$

$$\phi_{uv} \geqslant 0 \qquad \forall u,v \in \mathcal{U}_0 \tag{58}$$

where $\Gamma^+(i)$ and $\Gamma^-(o)$ are the successors of truck i and the predecessors of truck o on the handover graph, respectively.

The primal problem consists in minimizing the sojourn time (objective (51)) with end-start precedences given by the total order, start-start precedences between inbound and outbound trucks and time windows (constraints (52)). The dual problem is a maximum flow problem (objective (54)) where the difference between the outgoing and incoming flow equals the load received by the truck (constraints (55), (56) and (57)).

STRONG DUALITY According to the strong duality theorem and given a total order of the trucks on the doors, we can express the search for an optimal solution as a feasibility problem as only optimal solutions can satisfy the following system of primal-dual constraints:

$$\sum_{o\in\mathcal{O}} W_o s_o - \sum_{i\in\mathcal{I}} W_i s_i \geqslant \sum_{\substack{(u,v)\in\mathcal{U}\\ u\prec v}} \phi_{uv}.p_u + \sum_{v\in\mathcal{U}} \phi_{0v}.\bar{r}_v + \sum_{u\in\mathcal{U}} \phi_{u0}.(-\tilde{d}_u + p_u) \tag{59}$$

$$s_v - s_u \geqslant l_{uv} = \begin{cases} p_u & \forall u,v \in \mathcal{U}, u < v \\ 0 & \forall (u,v) \in \mathcal{A} \\ \overline{r_v} & \forall v \in \mathcal{U}, u = 0 \\ -\tilde{d}_u + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \tag{60}$$

$$\sum_{v\in\mathcal{U}} \phi_{v0} - \sum_{v\in\mathcal{U}} \phi_{0v} = 0 \tag{61}$$

$$\phi_{0i} - \phi_{i0} + \sum_{\substack{u\in\mathcal{U}\\ u\prec i}} \phi_{ui} - \sum_{\substack{v\in\mathcal{U}\\ i<v}} \phi_{iv} - \sum_{o\in\Gamma^+(i)} \phi_{io} = -W_i \qquad \forall i \in \mathcal{I} \tag{62}$$

$$\phi_{0o} - \phi_{o0} + \sum_{\substack{u\in\mathcal{U}\\ u\prec o}} \phi_{uo} + \sum_{i\in\Gamma^-(o)} \phi_{io} - \sum_{\substack{v\in\mathcal{U}\\ o\prec v}} \phi_{ov} = W_o \qquad \forall o \in \mathcal{O} \tag{63}$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \tag{64}$$

$$\phi_{uv} \geqslant 0 \qquad \forall u,v \in \mathcal{U}_0 \tag{65}$$

This system of constraints gather all the primal-dual constraints all together while the constraint (59) enforces the primal objective to be greater or equal to the dual objective (in fact they can only be equal for an optimal solution as the former is a lower bound of the latter, and vice-versa).

12.2.1.2 *Searching for the worst total ordering*

In a second step, the total order being no longer known, decision variables are introduced that model end-start precedences between trucks belonging to the same group, i.e.:

$$\forall g \in \mathcal{G}, \forall u, v \in g,\ \alpha_{uv} = \begin{cases} 1 & \text{if truck } u \text{ precedes truck } v \\ 0 & \text{otherwise} \end{cases}$$

The following constraints are added to the previous primal-dual inequation system:

$$s_v - s_u - p_u \geqslant (1 - \alpha_{uv}).(\overline{r_v} - \tilde{d_u}) \qquad \forall g \in \mathcal{G},\ \forall u, v \in g,\ u \neq v \qquad (66)$$

$$\alpha_{uv} + \alpha_{vu} = 1 \qquad \forall G \in \mathcal{G},\ \forall u, v \in G,\ u \neq v \qquad (67)$$

Constraints (66) ensure that, if there exists a end-start precedence between truck u and truck v, i.e. $\alpha_{uv} = 1$, then v cannot be started before the end of u. Constraints (67) imply that if trucks u and v belong to the same group and if truck u does not precede truck v then truck v precedes truck u, and vice versa.

Finally, the worst optimal sojourn time can be computed according to the following formulation:

$$\max \sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \qquad (68)$$

subject to

$$s_v - s_u \ \geqslant\ l_{uv} = \begin{cases} p_u & \forall u \in \mathcal{U},\ \forall v \in g^+(u) \\ 0 & \forall (u, v) \in \mathcal{A} \\ \bar{r}_v & \forall v \in \mathcal{U}, u = 0 \\ -\tilde{d}_u + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \qquad (69)$$

$$\sum_{v \in \mathcal{U}} \phi_{v0} - \sum_{v \in \mathcal{U}} \phi_{0v} = 0 \qquad (70)$$

$$\phi_{0i} - \phi_{i0} + \sum_{u \in g^-(i)} \phi_{ui} - \sum_{v \in g^+(i)} \phi_{iv} + \sum_{w \in g(i)} \alpha_{wi}.\phi_{wi} - \alpha_{iw}.\phi_{iw} - \sum_{o \in \Gamma^+(i)} \phi_{io} = -W_i \qquad \forall i \in \mathcal{I} \qquad (71)$$

$$\phi_{0o} - \phi_{o0} + \sum_{u \in g^-(o)} \phi_{uo} - \sum_{v \in g^+(o)} \phi_{ov} + \sum_{w \in g(o)} \alpha_{wo}.\phi_{wo} - \alpha_{ow}.\phi_{ow} + \sum_{i \in \Gamma^-(o)} \phi_{io} = W_o \qquad \forall o \in \mathcal{O} \quad (72)$$

$$\sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \leqslant \sum_{u \in \mathcal{U}} \phi_{0u}.\overline{r_u} + \sum_{u \in \mathcal{U}} \phi_{u0}.(p_u - \tilde{d_u}) + \sum_{g \in \mathcal{G}} \sum_{u,v \in g} \phi_{uv}.\alpha_{uv}.p_u + \sum_{u \in \mathcal{U}} \sum_{v \in g^+(u)} \phi_{uv}.p_u \quad (73)$$

$$s_v - s_u - p_u \geqslant (1 - \alpha_{uv}).(\overline{r}_v - \tilde{d}_u) \qquad \forall g \in \mathcal{G},\ \forall u, v \in g,\ u \neq v \quad (74)$$

$$s_v - s_u \geqslant p_u \qquad \forall u \in \mathcal{U},\ \forall v \in g^+(u) \quad (75)$$

$$\alpha_{uv} + \alpha_{vu} = 1 \qquad \forall g \in \mathcal{G},\ \forall u, v \in g,\ u \neq v \quad (76)$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \quad (77)$$

$$\phi_{uv} \geqslant 0 \qquad \forall u, v \in \mathcal{U} \quad (78)$$

$$\alpha_{uv} \in \{0, 1\} \qquad \forall u, v \in \mathcal{U} \quad (79)$$

All constraints and objectives have been already devised. Due to the introduction of α_{uv} variables, flow constraints (71), (72), and strong duality constraint (73) become non linear as they contain products of binary variables. Nevertheless, these constraints can be easily linearized using standard techniques so that a MIP solver can be used to solve the problem. The complementary variables and constraints that were used for linearization are detailed in the following.

12.2.1.3 *Worst optimal sojourn time problem linearization*

In order to linearize the problem, we introduce variables $\pi_{uv} = \alpha_{uv}.\phi_{uv}$ and M_{uv}, an upper bound of ϕ_{uv}.

$$\max \sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \quad (80)$$

subject to

$$s_v - s_u \geqslant l_{uv} = \begin{cases} p_u & \forall u \in \mathcal{U}, \forall v \in g^+(u) \\ 0 & \forall (u,v) \in \mathcal{A} \\ \overline{r_v} & \forall v \in \mathcal{U}, u = 0 \\ -\tilde{d_u} + p_u & \forall u \in \mathcal{U}, v = 0 \end{cases} \tag{81}$$

$$\sum_{v \in \mathcal{U}} \phi_{v0} - \sum_{v \in \mathcal{U}} \phi_{0v} = 0 \tag{82}$$

$$\phi_{0i} - \phi_{i0} + \sum_{u \in g^-(i)} \phi_{ui} - \sum_{v \in g^+(i)} \phi_{iv} + \sum_{w \in g(i)} (\pi_{wi} - \pi_{iw}) - \sum_{o \in \Gamma^+(i)} \phi_{io} = -W_i \quad \forall i \in \mathcal{I} \tag{83}$$

$$\phi_{0o} - \phi_{o0} + \sum_{u \in g^-(o)} \phi_{uo} - \sum_{v \in g^+(o)} \phi_{ov} + \sum_{w \in g(o)} (\pi_{wo} - \pi_{ow}) + \sum_{i \in \Gamma^-(o)} \phi_{io} = W_o \quad \forall o \in \mathcal{O} \tag{84}$$

$$\sum_{o \in \mathcal{O}} W_o s_o - \sum_{i \in \mathcal{I}} W_i s_i \geqslant \sum_{u \in \mathcal{U}} \phi_{0u}.\overline{r_u} + \phi_{u0}.(p_u - \tilde{d_u}) + \sum_{g \in \mathcal{G}} \sum_{\substack{u,v \in g \\ u \neq v}} \pi_{uv}.p_u + \sum_{u \in \mathcal{U}} \sum_{v \in g^+(u)} \phi_{uv}.p_u \tag{85}$$

$$s_v - s_u - p_u \geqslant (1 - \alpha_{uv}).(\overline{r_v} - \tilde{d_u}) \qquad \forall g \in \mathcal{G}, \forall u, v \in g,\ u \neq v \tag{86}$$

$$\alpha_{uv} + \alpha_{vu} = 1 \qquad \forall g \in \mathcal{G},\ \forall u, v \in g,\ u \neq v \tag{87}$$

$$\pi_{uv} \geqslant M_{uv}.(\alpha_{uv} - 1) + \phi_{uv} \qquad \forall g \in \mathcal{G}, \forall u, v \in g \tag{88}$$

$$\pi_{uv} \leqslant \phi_{uv} \qquad \forall g \in \}, \forall u, v \in g \tag{89}$$

$$\pi_{uv} \leqslant \alpha_{uv}.M_{uv} \qquad \forall g \in \}, \forall u, v \in g \tag{90}$$

$$s_u \in \mathbb{R} \qquad \forall u \in \mathcal{U} \tag{91}$$

$$\phi_{uv} \geqslant 0 \qquad \forall u, v \in \mathcal{U} \tag{92}$$

$$\alpha_{uv} \in \{0, 1\} \qquad \forall u, v \in \mathcal{U} \tag{93}$$

$$\pi_{uv} \geqslant 0 \qquad \forall g \in \mathcal{G}, \forall u, v \in g \tag{94}$$

12.2.2 *Worst maximum lateness*

This section is dedicated to the computation of the worst maximum lateness. We still consider the least favourable situation, i.e. the trucks arrive as late as possible. We are seeking for a total order offering the highest maximum lateness L_{max}.

The criterion for the worst maximum lateness, with the worst total order and the worst release dates is:

$$\max_{r} \max_{\theta} \min_{s_i, s_o} L_{max} \tag{95}$$

As in the case of the worst optimal sojourn time, one can still use a primal-dual formulation to determine the worst maximum lateness. Nevertheless, such an idea sounds ineffective as the worst maximum lateness can be computed in polynomial time. The proof can be found in the thesis (Le Gall, 1989) and holds for any regular time objective.

In the following section, we show how to find an optimal solution in polynomial time.

A sequence offering the worst optimal maximum lateness is illustrated in Figure 32, considering the same example as before.

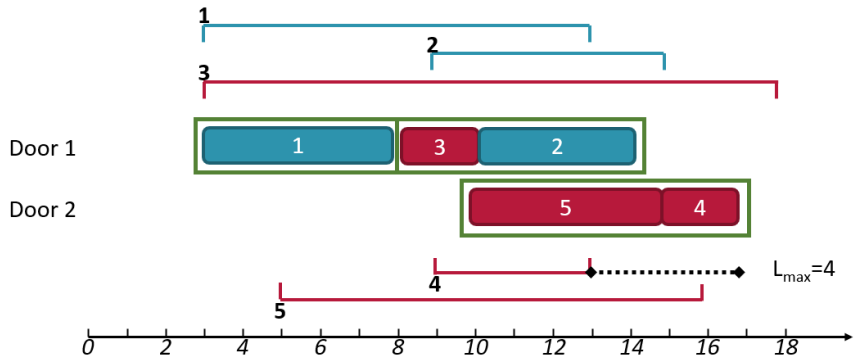

Figure 32: Worst maximum lateness

12.2.2.1 *Polynomial time computation*

The polynomial time computation of the worst maximum lateness considers the worst earliest start time $\overline{est}$ and the worst earliest finishing time $\overline{eft}$ of the truck handling tasks.

The worst earliest start time of a truck u depends on three different elements:

- the worst release date $\overline{r}_u$;
- the completion time of the last truck in the previous group $g^-(u)$;
- if u is an outbound truck, the starting time of inbound predecessors.

To calculate the worst earliest finishing time of a truck u, we consider the worst earliest finishing time over all trucks in the previous group, and the fact that truck u might be the last one of its group.

$$\overline{est_i} = \max\left(\max_{u \in g^-(i)} \overline{est_u} + \sum_{u \in g^-(i)} p_u \; ; \; \overline{r_i}\right) \qquad \forall i \in \mathcal{I} \qquad (96)$$

$$\overline{eft_i} = \max\left(\max_{\substack{v \in g(i) \\ v \neq i}} \overline{est_v} + \sum_{w \in g(i)} p_w \; ; \; \overline{est_i} + p_i\right) \qquad \forall i \in \mathcal{I} \qquad (97)$$

$$\overline{est_o} = \max\left(\max_{u\in g^-(o)} \overline{est_u} + \sum_{u\in g^-(o)} p_u \; ; \; \overline{r_o} \; ; \; \max_{i\in\Gamma^-(o)} \left(\overline{eft_i} - p_i\right)\right) \qquad \forall o \in \mathcal{O} \tag{98}$$

$$\overline{eft_o} = \max\left(\max_{\substack{v\in g(o)\\ v\neq o}} \overline{est_v} + \sum_{w\in g(o)} p_w \; ; \; \overline{est_o} + p_o\right) \qquad \forall o \in \mathcal{O} \tag{99}$$

Finally, to calculate the worst maximum lateness we take the greatest difference between the worst earliest finishing time and the due date over all trucks u.

$$L_{max} = \max_{u\in\mathcal{U}} \left(\overline{eft_u} - d_u\right) \tag{100}$$

12.2.2.2 *Resolution using Bellman - Ford algorithm*

Considering a specific truck u, the previous inequations can be easily modeled with a basic activity-on-node graph, according to the principles displayed in the partial graphs of the following figures. Figure 33 refers to the L_{max} for inbound trucks, whereas Figure 34 refers to the L_{max} for outbound trucks. Nodes 0 and 0^* are common for all trucks. L_{max} is obtained as the longest path from 0 to 0^*. The length of the arcs correspond to the formulas (96), (97), (98) and (99).

As such a graph does not hold any circuit, the circuit-free version of the Bellman-Ford algorithm can be used to solve this problem efficiently in $O(n \log n)$ (Cormen et al., 2001).

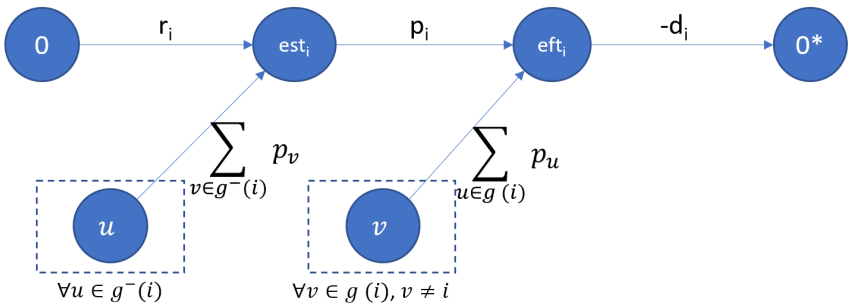

Figure 33: Graph around a vertex i

12.3 CREATION OF THE GROUP SEQUENCE

In order to test the evaluation process , a straightforward heuristic is proposed that only construct groups with either all inbound trucks

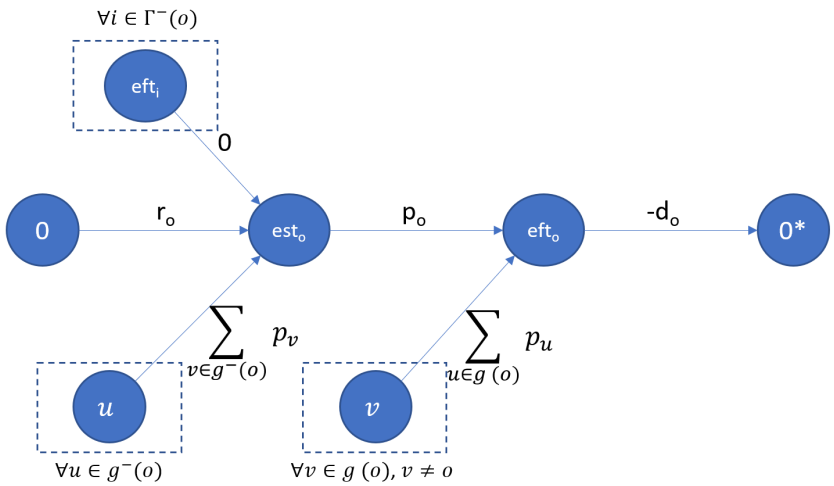

Figure 34: Graph around a vertex o

or all outbound trucks. This heuristic starts from a non-flexible feasible solution for the problem. Given the starting times of all the tasks (trucks), the assignment of trucks to doors is done by a MILP that minimizes the total number of successive inbound and outbound trucks on a same door.

Afterwards, for each door, the heuristic goes through the list of allocated trucks by increasing starting time and gathers inbound trucks together in one group and outbound trucks together in another group. Each time an outbound truck is encountered and added to a group, one need to check if the increasing group of inbound trucks contains a connected inbound truck. If so, the inbound group is added to the partial sequence. Whenever the size of a group reaches a predefined threshold, the group is added to the partial sequence as well, until all trucks are assigned to a group. A pseudo-code for this heuristic is presented in Algorithm 2.

12.4 PRELIMINARY EXPERIMENTS

In order to test the efficiency of the heuristic, we select the previous introduced set, with 250 instances and 5 doors, which allows us to work with enough doors to obtain relevant group sequences.

12.4.1 *Group sequence*

A parameter of the heuristic is the maximal size of the generated groups. We need to create groups with enough trucks inside to absorb by swapping possible perturbations, but limiting the group size is mandatory as too many trucks in one group will lead to highly deteriorated worst-case criteria values.

Algorithm 2 Group heuristic

Input: $\mathcal{S}$: feasible solution; f: group size
Output: Group sequence $\mathcal{G}$

for all door d **do**
  Sort the trucks allocated to door d in non-decreasing order of their starting time, provided by solution $\mathcal{S}$: $trucks_d$
  for all truck u in $trucks_d$ **do**
    if u is inbound **then**
      Put u in $list_i$
    else
      Put u in $list_o$
      if u has connection (precedences) with at least a truck in $list_i$ **then**
        Create a new group with the trucks in $list_i$, and add that group to the partial sequence
        Clear $list_i$
      end if
    end if
    if $size(list_i) \geqslant f$ **then**
      Create a new group with the trucks in $list_i$, and add that group to the partial sequence
      Clear $list_i$
    else if $size(list_o) \geqslant f$ **then**
      Create a new group with the trucks in $list_o$, and add that group to the partial sequence
      Clear $list_o$
    end if
  end for
  Create a new group with the trucks in $list_i$, and add that group to the partial sequence.
  Create a new group with the trucks in $list_o$, and add that group to the partial sequence
end for

max size	average	standard deviation
2	1.6	0.1
3	2.3	0.3
4	2.7	0.4

Table 28: Group size returned by the heuristic

Table 28 reflects some characteristics of the solutions provided by the heuristic for different maximal group sizes. The results are encouraging: a relatively simple and restrictive heuristic (where a group cannot contain both inbound and outbound trucks) succeeds in creating groups. With a maximal size of 2, we obtain groups of 1.6 trucks on average, meaning more than 2/3 of the trucks are permutable. Obviously, we cannot expect all trucks to be in a group of two trucks. When we increase the maximal size of the group to 3 and 4, we obtain respectively 2.3 and 2.7 trucks per group, on average. These numbers could be a good compromise between a non-flexible truck sequence (no permutation, but good objective values) and large groups (a lot of possible permutations, but poor worst-case values). We can mention the low standard deviation, meaning that the average size of the group is stable over the instances.

12.4.2 *Lateness*

As one of the criteria to evaluate the method, best and worst lateness can be calculated. We express the lateness values as a percentage of the time horizon: $\frac{L_{max}}{T}$.

We can see in Table 29 that the worst maximal lateness increases with the size of the groups. Having larger groups allows more permutations and thus also combinations with higher lateness, which can be obviously selected in the worst-case scenario. As expected, the worst-case value decreases as the time windows widen. With more feasible solutions, the worst maximal lateness is smaller. The number of trucks per door also has an impact. With more trucks per door, we increase the possible combinations thus the worst maximal lateness. We observe a large diversity of values, from 19% to 88%, meaning we probably even exceed sometimes the time horizon. A possible reason for these high values is that the group generation method is very basic and does not optimize the worst-case.

We can see in Table 30 that we obtain relevant results for the best maximal lateness: for all feasible instances, the percentages are negative. This means that we do not have any lateness and even some margin on the completion time of the worst truck. As a reminder, we mention the lateness value of the latest truck, which is obtained for

		max size		
trucks/door	TW	2	3	4
30		25%	42%	47%
40		32%	54%	66%
50		36%	55%	71%
70		40%	63%	78%
	0.00	46%	70%	88%
	0.25	49%	69%	83%
	0.50	43%	64%	76%
	0.75	25%	42%	57%
	1.00	19%	40%	53%
all	all	35%	55%	69%

Table 29: Worst maximal lateness relative to time horizon

the most favorable sequencing scenario with respect to the lateness criterion.

We also observe that the maximal group size does not affect the results significantly. We can nevertheless remark that for tight time windows, there is unsuspectingly no improvement, but this might be the result of a too-small computation time.

Considering lateness, we conclude that considering groups of 4 trucks might be irrelevant, as a single group of 4 trucks can ruin the worst-case value, due to the range of allowed permutations, and a group with a maximum of 3 trucks already brings sufficient flexibility. Anyhow, these experiments validate the efficient calculation of the worst and best lateness.

12.4.3 *Sojourn time*

Similar experiments are conducted with respect to the sojourn time criterion. As we cannot obtain easily the optimal objective value of all instances, we use the solution obtained after 2h of computation in chapter 8 as the reference solution value. We express the results as the relative difference regarding this value.

Looking at the worse case in Table 31, the average differences with the good-quality solutions are high, from 263 to 346%. Of course, the solution value of the worst case is a lot worse than the solution value of the good solution, as the good solution tries to find the best possible solution. Having larger groups allows more permutations, which naturally implies higher sojourn time in the worst case. We observe that the number of trucks per door does not really cause a signifi-

		max size		
trucks/door	TW	2	3	4
30		-17%	-17%	-18%
40		-14%	-15%	-15%
50		-15%	-15%	-15%
70		-11%	-11%	-11%
	0.00	0%	0%	0%
	0.25	0%	0%	0%
	0.50	-5%	-6%	-6%
	0.75	-24%	-24%	-24%
	1.00	-30%	-30%	-30%
all	all	-14%	-14%	-14%

Table 30: Best maximal lateness relative to time horizon

cant variation. However, the tightness of the time windows is more impacting. Limiting the creation of groups thus limits the number of possible permutations and the range of possible schedules. These high values also come from the fact that our simple heuristic aims to create a large group size and does not consider the worst-case sojourn time.

The last Table 32, with very low differences between the best case in sojourn time, shows that the evaluation works properly: within the 15 minutes of computation time allocated to each evaluation, the solutions obtained are really good. The different percentages that can be read in the Table are not that different from each other. A good solution is used to start the algorithm, and both the time window tightness and the number of trucks per door slightly influence the final solution quality. Enlarging the time windows leads mainly to solutions with bigger time laps between inbound and outbound, increasing the sojourn time.

12.4.4 *Preliminary experiments conclusion*

This first set of experiments returns results with relatively high values, but this is acceptable because of the rather poor performance of the group creation heuristic. These experiments validate the ability of this heuristic to create group sequences, and the ability of the evaluation methods to calculate the best and worst-case values of both objectives.

Obviously, the work on both the evaluation and the creation of groups and group schedules should be continued. Several points have to be developed. The appropriate computing time limit for each evalu-

		max size		
trucks/door	TW	2	3	4
30		265%	278%	319%
40		240%	282%	307%
50		279%	345%	387%
70		284%	323%	364%
	0.00	148%	198%	253%
	0.25	212%	249%	274%
	0.50	304%	354%	414%
	0.75	263%	328%	366%
	1.00	325%	346%	368%
all	all	263%	307%	346%

Table 31: Worst sojourn time relative to CP solution

		max size		
trucks/door	TW	2	3	4
30		5%	7%	6%
40		6%	6%	6%
50		6%	7%	6%
70		7%	7%	7%
	0.00	1%	1%	1%
	0.25	4%	5%	5%
	0.50	12%	14%	12%
	0.75	6%	6%	6%
	1.00	5%	5%	5%
all	all	6%	7%	6%

Table 32: Best sojourn time relative to CP solution

ation criterion should be determined. The evaluation method should be tested on larger instances as well. And finally, an efficient methodology should be developed to create appropriate group sequences.

13

CONCLUSION AND FUTURE WORKS

In this thesis, the crossdock truck scheduling problem has been studied in order to favor an efficient synchronization of the trucks as well as to promote a fluid rotation of the goods inside the crossdock. Those are both performance indicators that are frequently considered by supply chain managers. The considered problem consists in assigning trucks to doors as well as scheduling trucks on the different doors in order to optimize an objective function. Minimizing the total sojourn time of the pallets in the storage has been considered as the objective as it reflects both trucks synchronization and goods turnover.

Considering first a deterministic version of this scheduling problem where each truck is characterized by its handling time, its time windows, its processing time, its load and a set of handover relations (which model the pallet flows), we study its complexity under various assumptions. Such problem is NP-hard in its general version (as finding a feasible solution respecting the time windows is NP-complete). Nonetheless, we show that even when time windows are relaxed, the problem remains NP-hard if at least two doors are considered. For the very restrictive single-door case, CTSP becomes polynomial when the handover graph can be partitioned in bicliques; otherwise its status is open.

We also provide a set of 1000 instances with up to 10 doors, which will be made available for the scheduling community. Designed to impose a 65% workload inside the crossdock, they can be seen as realistic from an industrial viewpoint. Of course, since real crossdocks may contain hundreds of doors in practice, this claim must be somewhat mitigated. Anyway, as observed during our computational experiments, our instances seem to be complex enough to reach the limits of the resolution capabilities of exact solving methods.

Another contribution of this thesis lies in the proposal of different exact methods that all have been explained and evaluated on our instance sets. In particular, we propose a branch-and-cut algorithm that exploits the concept of critical sets of tasks, well known in scheduling research. Instead of focusing only on minimal critical sets as other methods, we show that extending to non-minimal critical sets can be interesting. Indeed, it allows to exhibit strong valid inequalities that can be used to progressively converge to an optimal solution, by iteratively integrating cuts in the problem. Since potentially, there is an exponential number of cuts, we show how cuts can be efficiently separated from one iteration to another. Note that our cut-separation approach and our family of valid inequalities can be used in schedul-

ing environments other than CTSP, which also handle cumulative resources. This branch-and-cut approach has been implemented in different ways, sometimes integrating MIP and CP solvers. All approaches have been evaluated and the results are quite mitigated. For pure CP formulations, IBM CPO performs very well in finding feasible solutions but fails to find good lower bounds. Conversely, our branch-and-cut methods are not very good at finding feasible solutions but can provide better lower bounds. In any case, the ability of the methods to find optimal solutions quickly becomes disappointing as the number of doors increases.

Our last contribution concerns a non-deterministic version of our truck scheduling problem where the arrival dates of trucks are uncertain, and modeled as equiprobable time intervals. We sketch a proactive-reactive solution method that uses the concept of groups of permutable trucks. As those groups are sequenced and assigned to specific doors through the proactive phase, a reactive scheduling algorithm is able to dynamically determine the order of the trucks within each group, based on the actual realization of the arrival dates. Considering a specific group sequence, we discuss how it can be evaluated with respect to total sojourn time and maximum lateness. Four different decision problems have been designed to compute the best and worst sojourn time and the best and worst maximum lateness, respectively. We prove that all of these problems are NP-hard, even when the group sequence is given, except for the worst maximum delay which remains polynomial. We show how the other three objectives can be computed either by a CP or MIP approach.

There are many perspectives to our work. Concerning the complexity, a first perspective is to settle the complexity status of CTSP(1,C) that remains open for general handover graph. Addressing the complexity of the problem in different crossdocking scenarios (e.g. storage restrictions, exclusive-mode doors etc) could also be of interest.

Clearly, better exact methods need to be designed to optimally solve real-size CTSP instances. Many avenues are open for this purpose. First, in the short term, we would like to better explore the influence of the threshold that triggers the addition of new cuts in our branch-and-cut algorithm. We would also like to incorporate some dominance rules from the literature into our approach and design new ones. Indeed, such rules could restrict the search space further and make our method more efficient. Of course, more sophisticated branch-and-cut or branch-and-cut and price methods could also be designed in the longer term. Another possibility could be to find better hybridizations of the MIP and CP solvers as they already incorporate many very efficient algorithms. Finally, the development of heuristics, metaheuristics or matheuristics for solving CTSP is also an interesting line of research as these methods can provide good upper bounds for exact methods.

Regarding the non-deterministic version of the CTSP, we still need to upgrade our groups generation method, in order to optimize the criteria of worst and best cases for our objectives. We also need to evaluate experimentally how some reactive scheduling rules can take profit of group sequences. Another research direction will be to evaluate the quality of group sequences with respect to the worst response of a reactive algorithm, instead of considering the worst case. Indeed, the worst case evaluation is known to be too conservative. We do not investigate procedures able to compute efficient group sequences. Designing such procedures, able to take the nature of the reactive algorithm into account, will be ideal. Finally, more ad-hoc proactive-reactive methods could be designed that take more accurately the nature of the CTSP into account. Indeed, only the trucks that are approaching the crossdock have an uncertain release date, while the others have a planned release date. The interval of uncertainty evolves dynamically as the truck gets closer to the crossdock. Therefore, the robustness could only concentrate on those trucks, instead of considering all the trucks at once as we did.

BIBLIOGRAPHY

Alpan, G., R. Larbi, and B. Penz (2011). "A bounded dynamic programming approach to schedule operations in a cross docking platform." In: *Computers & Industrial Engineering* 60 (3), pp. 385–396.

Alvarez-Pérez, G.A., J.L. González-Velarde, and J. W. Fowler (2009). "Crossdocking Just in Time scheduling: An alternative solution approach." In: *Journal of the Operational Research Society* 60 (4), pp. 554–564.

Amini, A. and R. Tavakkoli-Moghaddam (2016). "A bi-objective truck scheduling problem in a cross-docking center with probability of breakdown for trucks." In: *Computers & Industrial Engineering* 96, pp. 180–191.

Apte, M. and S. Viswanathan (2000). "Effective Cross Docking for Improving Distribution Efficiencies." In: *International Journal of Logistics Research and Applications* 3 (3), pp. 291–302.

Artigues, C., J.-C. Billaut, A. Cheref, N. Mebarki, and Z. Yahouni (2016). "Robust Machine Scheduling Based on Group of Permutable Jobs." In: *Robustness Analysis in Decision Aiding*. Vol. 241. Optimization, and Analytics. International Series in Operations Research & Management Science. Springer, pp. 191–220.

Artigues, C., J.-C. Billaut, and C. Esswein (2005). "Maximization of solution flexibility for robust shop scheduling." In: *European Journal of Operational Research* 165, 314–328.

Ashtiani, B., R. Leus, and MB. Aryanezhad (2011). "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing." In: *Journal of Scheduling* 14, 157–171.

Assadi, M. T. and M. Bagheri (2016). "Scheduling trucks in a multiple-door cross docking system with unequal ready times." In: *J. Industrial Engineering* 10 (1), pp. 103–125.

Bartholdi, J. and K. Gue (2004). "The best shape for a crossdock." In: *Transportation Science* 38 (2), pp. 235–244.

Berghman, L., C. Briand, R. Leus, and P. Lopez (2015). "The truck scheduling problem at crossdocking terminals: Exclusive versus mixed mode." In: *ICORES 2015, Proceedings*, pp. 247–253.

Billaut, J.-C., A. Moukrim, and E. Sanlaville (2008). *Flexibility and Robustness in Scheduling*. John Wiley Sons, Ltd.

Bodnar, P., R. de Koster, and K. Azadeh (2015). "Scheduling Trucks in a Cross-Dock with Mixed Service Mode Dock Doors." In: *Transportation Science* 51 (1), pp. 112–131.

Boloori Arabani, A.R., S. M. T. Fatemi Ghomi, and M. Zandieh (2010). "A multi-criteria cross-docking scheduling with just-in-time approach." In: *International Jounal of Advanced Manufacturing Technology* 49 (5-8), pp. 741–756.

Boloori Arabani, A.R., S. M. T. Fatemi Ghomi, and M. Zandieh (2011a). "Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage." In: *Expert Systems with Applications* 38 (3), pp. 1964–1979.

Boloori Arabani, A.R., S. M. T. Fatemi Ghomi, and M. Zandieh (2012). "A cross-docking scheduling problem with sub-population multi-objective algorithms." In: *The International Journal of Advanced Manufacturing Technology* 58 (5-8), pp. 741–761.

Boloori Arabani, A.R., M. Zandieh, and S.M.T. Fatemi Ghomi (2011b). "Multi-objective genetic-based algorithms for a cross-docking scheduling problem." In: 11 (8), pp. 4954–4970.

Bourreau, E., M. Gondran, P. Lacomme, and M. Vinot (2019). *De la programmation linéaire à la programmation par contraintes*. Ellipses.

Boysen, N. (2010). "Truck scheduling at zero-inventory cross docking terminals." In: *Computers & Operations Research* 37 (1), pp. 32–41.

Boysen, N., D. Briskorn, S. Fedtke, and M. Schmickerath (2019). "Automated sortation conveyors: A survey from an operational research perspective." In: *European Journal of Operational Research* 276 (3), pp. 796–815.

Boysen, N. and M. Fliedner (2010). "Cross dock scheduling: Classification, literature review and research agenda." In: *Omega* 38 (6), pp. 413–422.

Briskorn, D., M. Fliedner, and M. Tschöke (2021). "Vehicle Sequencing at Transshipment Terminals with Handover Relations." In: *INFORMS Journal on Computing* 33.2 (September), pp. 477–494.

Briskorn, D., F. Jaehn, and E. Pesch (2013). "Exact algorithms for inventory constrained scheduling on a single machine." In: *Journal of Scheduling* 16 (1), 105–115.

Buijs, P., I. Vis, and H. Carlo (2014). "Synchronization in Cross-Docking Networks: A Research Classification and Framework." In: *European Journal of Operational Research* 239, 593–608.

Carrera, S., K. Chami, R. Guimaraes, M.C. Portmann, and W. Ramdane-Cherif (2008). *Negotiation models for logistic plarform planning and scheduling*.

Chen, R., B. Fan, and G. Tang (June 2009). "Scheduling Problems in Cross Docking." In: *Combinatorial Optimization and Applications: COCOA 2009, Proceedings*. Vol. 5573, pp. 421–429.

Christofides, N., R. Alvarez-Valdes, and J.M. Tamarit (1987). "Project scheduling with resource constraints: A branch-and-bound approach." In: *European Journal of Operational Research* 29 (3), pp. 262–273.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and S. Clifford (2001). In: *Introduction To Algorithms (2nd ed.)* P. 978.

Demeulemeester, E. and W. Herroelen (2002). *Project scheduling: a research handbook*. Vol. 49. International Series in Operations Research Management Science. Springer Science & Business Media.

Diglio, A., A. Genovese, and P. Carmela (2017). *An Optimization Model for the Outbound Truck Scheduling Problem at Cross-Docking Platforms*. Ed. by Springer. Vol. 217, pp. 601–610.

Duan, Y., Y. O. Yao, X. Zhang, and J. Huo (2020). "Unraveling Cross Docking : Operations Performance , Demand Uncertainty and Fulfillment Mechanism." In: *SSRN Electronic Journal*, pp. 1–41.

Dyer, M. E. and L. A. Wolsey (1990). "Formulating the single machine sequencing problem with release dates as a mixed integer problem." In: *Discrete Applied Mathematics* 26 (2-3).

Fabry, Q., A. Agnetis, L. Berghman, and C. Briand (2021a). "On the complexity of the crossdock truck-scheduling problem." In: *17th International Workshop on Project Management and Scheduling*, pp. 188–195.

Fabry, Q., A. Agnetis, L. Berghman, and C. Briand (2022). "Complexity of flow time minimization in a crossdock truck scheduling problem with asymmetric handover relations." In: *Operations Research Letters* 50.1, pp. 50–56.

Fabry, Q., L. Berghman, and C. Briand (2021b). "Using Flexible Crossdock Schedules to Face Truck Arrivals Uncertainty." In: *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems*. Springer International Publishing, pp. 188–195.

Fazel Zarandi, M.H., H. Khorshidian, and M. Akbarpour Shirazi (2016). "A constraint programming model for the scheduling of JIT cross-docking systems with preemption." In: *Journal of Intelligent Manufacturing* 27 (2), pp. 297–313.

Fedtke, S. and N. Boysen (2017). "Layout planning of sortation conveyors in parcel distribution centers." In: *Transportation Science* 51 (1), pp. 3–18.

Forger, G. (1995). "UPS starts world's premiere cross-docking operation." In: *Modern material handling*, pp. 36–38.

Galbreth, M., J. Hill, and S. Handley (2008). "An Investigation of the Vamue of Cross-docking for Supply Chain Management." In: *Journal of Business Logistics* 29 (1), pp. 225–239.

Garey, M. and D. Johnson (1978). "Strong NP-completeness results: Motivation, examples, and implications." In: *Journal of the ACM* 25.3, pp. 499–508.

Gaudioso, M., M. F. Monaco, and M. Sammarra (2020). "A Lagrangian heuristics for the truck scheduling problem in multi-door, multi-product Cross-Docking with constant processing time." In: *Omega (United Kingdom)*, p. 102255.

Golias, M. M., G. K. D. Saharidis, and H. E. Haralambides (2013). "Advances in Truck Scheduling at a Cross Dock Facility." In: *International Journal of Information Systems and Supply Chain Management* 6 (3), pp. 40–62.

Guo, P., F. Weidinger, and N. Boysen (2019). "Parallel machine scheduling with job synchronization to enable efficient material flows in hub terminals." In: *Omega* 89, pp. 110–121.

Hedler-Staudt, F., G. Alpan, M. Di Mascolo, and C. M. Rodriguez (2015). "Warehouse performance measurement: a literature review." In: *International Journal of Production Research* 53 (18), pp. 5524–5544.

Heidari, F., S. H. Zegordi, and R. Tavakkoli-Moghaddam (2018). "Modeling truck scheduling problem at a cross-dock facility through a bi-objective bi-level optimization approach." In: *Journal of Intelligent Manufacturing* (5), pp. 1155–1170.

Hermel, D., H. Hasheminia, N. Adler, and M. Fry (2016). "A solution framework for the multi-mode resource-constrained cross-dock scheduling problem." In: *Omega* 59, pp. 157–170.

Konur, D. and M. M. Golias (2013a). "Analysis of different approaches to cross-dock truck scheduling with truck arrival time uncertainty." In: *Computers & Industrial Engineering* 65 (4), pp. 663–672.

Konur, D. and M. M. Golias (2013b). "Cost-stable truck scheduling at a cross-dock facility with unknown truck arrivals: A meta-heuristic approach." In: *Transportation Research Part E: Logistics and Transportation Review* 49 (1), pp. 71–91.

Kouvelis, P. and G. Yu (1997). *Robust Discrete Optimization and Its Applications*. Springer US.

Ladier, A.-L. and G. Alpan (2013). "Scheduling truck arrivals and departures in a crossdock: Earliness, tardiness and storage policies." In: *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*.

Ladier, A.L. (2015). "Planification des opérations de crossdocking."

Ladier, A.L. and G. Alpan (2016a). "Cross-docking operations: Current research versus industry practice." In: *Omega (United Kingdom)* 62, pp. 145–162.

Ladier, A.L. and G. Alpan (2016b). "Robust cross-dock scheduling with time windows." In: *Computers and Industrial Engineering* 99, pp. 16–28.

Ladier, A.L. and G. Alpan (2018). "Crossdock truck scheduling with time windows: earliness, tardiness and storage policies." In: *Journal of Intelligent Manufacturing* 29 (3), pp. 569–583.

Larbi, R., G. Alpan, P. Baptiste, and B. Penz (2011). "Scheduling cross docking operations under full, partial and no information on inbound arrivals." In: *Computers and Operations Research* 38 (6), pp. 889–900.

Le Gall, A. (1989). "Un système interactif d'aide à la décision pour l'ordonnancement et le pilotage en temps réel d'atelier." LAAS-CNRS.

Le Pape, C. (2014). "Constraint Programming." In: *Concepts of Combinatorial Optimization*, pp. 325–338.

Lenstra, J.K., A.H.G. Rinnooy Kan, and P. Brucker (1977). "Complexity of machine scheduling problems." In: *Annals of Discrete Mathematics* 1, pp. 343–362.

Li, Y., A. Lim, and B. Rodrigues (2004). "Crossdocking - JIT scheduling with time windows." In: *Journal of the Operational Research Society* 55 (12), pp. 1342–1351.

Maknoon, M.Y. (2013). "Scheduling material handling in cross-docking terminals." École polytechnique de Montréal.

Maknoon, M.Y. and P. Baptiste (2009). "Cross-docking: increasing platform efficiency by sequencing incoming and outgoing semi-trailers." In: *International Journal of Logistics: Research and Applications* 12 (4), pp. 249–261.

Monaco, F. and M. Sammarra (2019). "A New Formulation of the Single Door Truck Scheduling Problem." In: Springer, Cham, pp. 291–301.

Napolitano, M. (2011). "Cross dock fuels growth at Dots." In: *Logistics Management* 50(2), pp. 30–34.

Neufeld, J.S., J. N. D. Gupta, and U. Buscher (2016). "A comprehensive review of flowshop group scheduling literature." In: *Computers & Operations Research* 70, pp. 56–74.

Patterson, T. (2018). *2018 Distribution Network Trends Report*, pp. 1–4.

Pinedo, M. L. (2012). *Scheduling*. Vol. 29. Springer.

Pinot, G. and N. Mebarki (2009). *An exact method for the best case in a group sequence: Application to a general shop problem*, pp. 1269–1274.

Porter, M. (2008). "The five competitive forces that shape strategy." In: *Harvard Business Review* 86, pp. 78–93, 137.

Potts, C.N. (1980). "An algorithm for the single machine sequencing problem with precedence constraints." In: *Mathematical Programming Studies*, pp. 13, 78–87.

PwC Report, *NA* (2013). *Next-generation supply chains: Efficient, fast and tailored*.

Rajabi, M. and M. A. Shirazi (2016). "Truck scheduling in a cross-dock system with multiple doors and uncertainty in availability of trucks." In: *Journal of Applied Environmental and Biological Science* 6, pp. 101–109.

Rijal, A., M. Bijvank, and R. De Koster (2016). "Integrated Versus Sequential Scheduling and Assignment at a Unit Load Cross-dock." In: *IMHRC*, p. 23.

Rijal, A., M. Bijvank, and R. De Koster (2019). "Integrated scheduling and assignment of trucks at unit-load cross-dock terminals with

mixed service mode dock doors." In: *European Journal of Operational Research* 278 (3), pp. 752–771.

Sadykov, R. (2012). "Scheduling incoming and outgoing trucks at cross docking terminals to minimize the storage cost." In: *Annals of Operations Research* 201 (1), pp. 423–440.

Serrano, C., X. Delorme, and A. Dolgui (2017). "Scheduling of truck arrivals, truck departures and shop-floor operation in a cross-dock platform, based on trucks loading plans." In: *International Journal of Production Economics* 194, pp. 102–112.

Shakeri, M., M.Y.H Low, S. Turner, and E.W. Lee (2012). "A robust two-phase heuristic algorithm for the truck scheduling problem in a resource-constrained crossdock." In: *Computers and Operations Research* 39, pp. 2564–2577.

Stalk, G., P. Evans, and L.E. Shulman (1992). "Competing on capabilities: the new rules of corporate strategy." In: *Harvard Business Review* 70 (2), pp. 57–69.

Tadumadze, G., N. Boysen, S. Emde, and F. Weidinger (2019). "Integrated truck and workforce scheduling to accelerate the unloading of trucks." In: *European Journal of Operational Research* 278 (1), pp. 343–362.

Tsui, L. Y. and C.-H. Chang (1990). "A microcomputer based decision support tool for assigning dock doors in freight yards." In: *Computers & Industrial Engineering* 19.1-4, pp. 309–312.

Van Belle, J., P. Valckenaers, and D. Cattrysse (2012). "Cross-docking: State of the art." In: *Omega* 40 (6), pp. 827–846.

Van de Vonder, S., F. Ballestín, E. Demeulemeester, and W. Herroelen (2007). "Heuristic procedures for reactive project scheduling." In: *Computers & Industrial Engineering* 52 (1), pp. 11–28.

Waller, M., R. Cassady, and J. Ozment (2006). "Impact of cross-docking on inventory in a decentralized retail supply chain." In: *Transportation Research Part E: Logistics and Transportation Review* 42 (5), pp. 359–382.

Witt, C.E. (1998). "Crossdocking: Concepts demand choice." In: *Material handling engineering* 53(7), pp. 44–49.

Wolsey, L. (1998). *Integer Programming*. Wiley-Interscience.

Wu, S.D., E.S. Byeon, and R.H. Storer (1999). "A Graph-Theoretic Decomposition of the Job Shop Scheduling Problem to Achieve Scheduling Robustness." In: *Operations Research* 47.1, pp. 113–124.

Xi, X., L. Changchun, W. Yuan, and L. Loo Hay (2020). "Two-stage conflict robust optimization models for cross-dock truck scheduling problem under uncertainty." In: *Transportation Research Part E: Logistics and Transportation Review* 144, pp. 102–123.

Yahouni, Z. (2019). "Le meilleur des cas pour l'ordonnancement de groupes : Un nouvel indicateur proactif-réactif pour l'ordonnancement sous incertitudes."

Ye, Y., J. F. Li, R.Y.K. Fung, K. Li, and H. Fu (2018). "Optimizing truck scheduling in a cross-docking system with preemption and unloading/loading sequence constraint." In: *ICNSC 2018 - 15th IEEE International Conference on Networking, Sensing and Control*, pp. 1–6.

Yu, W. and Egbelu J. (2008). "Scheduling of inbound and outbound trucks in cross docking systems with temporary storage." In: *International Journal of Production Economics* 184 (1), pp. 377–396.

Zeballos, L.J. (2010). "A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems." In: *Robotics and Computer-Integrated Manufacturing* 26.6, pp. 725–743.

Zouhaier, H., L. Ben Said, S. Sboui, and L. Bardo (2016). *Real-world Truck Scheduling for Cross-dock Operations : an alternative solution approach Literature review*.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/