CROSS-DOCK DOOR ASSIGNMENTS: MODELS, ALGORITHMS AND EXTENSIONS

WAEL NASSIEF

A THESIS

IN

THE DEPARTMENT

OF

MECHANICAL AND INDUSTRIAL ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY (INDUSTRIAL ENGINEERING)

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

MAY, 2017

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Wael Nassief**

Entitled: **Cross-dock door assignments: models, algorithms & extensions**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Industrial Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Charalambos Poullis	Chair
Dr. Iris F. A. Vis	External Examiner
Dr. Ali Akgunduz	Examiner
Dr. Jia Yuan Yu	Examiner
Dr. Lata Narayanan	Examiner
Dr. Ivan Contreras	Supervisor
Dr. Brigitte Jaumard	Supervisor

Approved by

(Chair of Department or Graduate Program Director)

May 2017

(Dean of Engineering and Computer Science Faculty)

ABSTRACT

Cross-dock door assignments: models, algorithms and extensions

Wael Nassief

Doctor of Philosophy in Industrial Engineering

Concordia University, 2017

In a cross-dock, goods coming from numerous origins get unloaded from incoming trucks, consolidated according to their destinations, and then loaded into outgoing trucks with little or no storage in between. We study a class of cross-dock door assignment problems where the assignments of origins (or incoming trucks) to inbound doors, and destinations (or outgoing trucks) to outbound doors are determined with the objective of minimizing the handling cost. Cross-dock door assignment problems are a fundamental class of optimization problems in cross-docking as they arise in more complex operational problems incorporating other decisions such as scheduling, routing, and workforce allocation.

We first introduce several linear mixed integer programming formulations with Lagrangean relaxation and column generation algorithms based on some of these formulations. We then theoretically and computationally compare these formulations in terms of their linear, Lagrangean and combinatorial relaxations. Finally, we integrate the assignments with sequencing and selection decisions, based on our observations on a large cross-dock company in the USA, and introduce two new integer programming formulations. Where possible, our work is compared with existing ones, and new sets of instances are generated to either vary or enlarge the current data sets in the literature.

Acknowledgments

I am grateful to all the people who have been part of my PhD journey throughout the last 4 years. First and foremost, my supervisors Dr. Ivan Contreras and Dr. Brigitte Jaumard for their continuous advices, support, teaching and training throughout these years. I have very much enjoyed working with you two and learned a great deal of knowledge and experience during this time. I am so thankful and grateful to Ivan. Thank you for being patient, wise and endlessly giving during this journey. I would not have imagined completing my PhD without your support. My gratitude also goes to Brigitte for your teaching and support year after year. I am so happy to have met you and worked with you in this journey. I am so grateful to all the tips and experiences you have taught me. Special thanks to Dr. Rami As'ad for introducing me to Ivan in the first year of my PhD and to help me start this journey.

I am forever grateful to Dr. Monique Guignard and Dr. Peter Hahn along with the department of Operations, Information and Decisions at Wharton School in Philadelphia for their generosity in hosting me for four months during my research visit. Monique, thank you for your warm welcome, and care, and knowledge during my visit. Peter, thank you for your support and guidance in facilitating the visits and meetings with the Cross-dock facility in the USA. I have learned a lot from both of you and enjoyed working with you two.

To all my friends, thank you for your love and support during these years. Special gratitudes towards my friends Christian, Sonia, Furkan and my mentor and dear friend Jerry, and of course, my beloved partner whom has given me an endless, unconditional love through these years. You are my family. My colleagues and lab mates, thank you for putting up with me throughout these years. Thank you for the great support and discussions that we have had on the white board of our lab. It has been a pleasure working with you all day after day. To all my Saudi and Moroccan families: thank you for your prayers.

Finally, I am grateful to the Higher Education Ministry in Saudi Arabia for their financial support and funding of my PhD.

Contribution of Authors

This dissertation is presented in a manuscript-style. It contains three articles that have been either accepted for publication or are currently under revision in different journals. They are presented here as follows. The first article entitled "A mixed-integer programming formulation and Lagrangean relaxation for the cross-dock door assignment problem" was published in February 2016 in the *International Journal of Production Research* and co-authored with Dr. Ivan Contreras and Dr. Rami As'ad. The second article entitled "A comparison of formulations and relaxations for the cross-dock door assignment problems" was submitted in March 2017 to the journal of *Computers and Operations Research*, and is co-authored with Dr. Ivan Contreras and Dr. Brigitte Jaumard. The third article entitled "The container scheduling and cross-dock door selection problem" was submitted in March 2017 to the *International Journal of Production Research*, and is co-authored with Dr. Ivan Contreras, Dr. Monique Guignard, Dr. Peter Hahn, and Dr. Brigitte Jaumard.

The author of this thesis acted as the principal researcher with the corresponding duties of developing mathematical formulations, proofs, developing and implementing the algorithms and analysis of computational experiments along with the writing of the first drafts of all articles.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Distribution plays an important role, in any supply chain network, by transferring products from origins to destinations, directly or passing through consolidation centers, while adopting some form of logistics strategies throughout the whole chain. One of these strategies is *cross-docking*, where goods are unloaded from incoming trucks, consolidated according to their destinations, and then, loaded into outgoing trucks with little or no storage in between. This process takes place at cross-docking facilities that consist of *strip* doors for unloading, *stack* doors for loading and a sorting area in between allowing for an appropriate transshipment that is normally carried out by employees and material handling equipment. Cross-docking, in its best practice, aims to reduce or ultimately eliminate two major functions of a traditional warehouse: *storage* and *order picking*, assuming that relevant environmental settings are met. Cross-docking was first introduced during the 1930s by the US trucking industry according to Arnaout et al. [8], and then in the retailing industry by Wal-Mart since the 1980s (Stalk et al. [54]).

As of today, cross-docking has gained global recognition by companies and researchers. This strategy has been reportedly used in the retailing, manufacturing, automotive, and photographic industries. For examples of successful cross-docking implementations, the interested reader is referred to Forger [25], Kinnear [31], Witt [60], Chen and Song [16], and Napolitano [41]. Research-wise, more than 85% of total publications in cross-docking have appeared after 2004. Given the inherent complexity of designing and operating cross-docks, several classes of decision problems have been studied in the literature. Agustina et al. [2], Boysen [10], Shuib and Fatthi [53], Van Belle et al. [58], and Buijs et al. [14] provide reviews of decision problems arising in cross-docking.

Moreover, a recent review by Ladier and Alpan [32] proposes a framework that highlights the gaps between the literature and some cross-docking practices in France.

Cross-docking works effectively only for products with specific features. Apte and Viswanathan [7] consider *product demand rate* and *unit stock-out* cost as the most important factors that contribute to a suitable application of cross docking. The former, when stable, makes it easier to predict and coordinate the products flow in cross-docking while the latter is preferable when it is low because the possibility of running out of products is higher due to cross-docking nature. Figure 1.1 illustrates the suitability of cross-docking as found in Van Belle et al. [58]. These two

		Product demand rate	
		Stable and constant	Unstable or fluctuating
Unit stock-out costs	High	Cross-docking can be implemented with proper systems and planning tools	Traditional distribution preferred
	Low	Cross-docking preferred	Cross-docking can be implemented with proper systems and planning tools

Figure 1.1: Suitability of Cross-docking Apte and Viswanathan [7].

guidelines lead to a range of suitable products that are reported in the literature, which are: durable products, high value and high security, perishable, and temperature-controlled products (Van Belle et al. [58], Apte and Viswanathan [7], Saddle Creek Corporation [50]). Generally, fast moving goods with fairly constant demands are the best candidates for cross-docking as seen in Ross and Jayaraman [49].

From an operations research point of view, there is a vast array of challenging optimization problems that need to be solved in cross-docking, ranging from strategic, tactical to operational decision problems, and more importantly, there is a spacious room for improvement. These optimization problems can be classified, according to Van Belle et al. [58], in seven general categories. The *location* of cross-docks and their *layout* optimization are two classes of strategic decision problems. Tactically, *network* design problems focus on determining the optimal flow of goods through a single or multiple cross-docks. Operational decision problems include *vehicle routing*, *dock door assignment*, *truck scheduling*, and *temporary storage* problems.

In this thesis we study a fundamental class of cross-docking operational problems referred to as *cross-dock door assignment problems* (CDAPs), where the assignments of incoming trucks to inbound doors and outgoing trucks to outbound door is determined with the objective of minimizing the total handling cost. We then integrate the assignments with the decisions of sequencing trucks and selecting dock doors to process the trucks in a given planning period. The introduction of these interdependent decisions are practically driven from our observations on a large cross-dock facility in the USA. The objectives of this thesis can be summarized as follows:

- To introduce new mathematical formulations for the CDAP.
- To introduce new combinatorial, linear and Lagrangean relaxations for the CDAP.
- To compare these formulations and relaxations computationally and theoretically.
- To design approximate and exact algorithms that act as bounding procedures on the CDAP.
- To extend the CDAP by integrating its assignment decisions with sequencing and selection decisions, and introduce mathematical programming formulations for these extensions.

This thesis is composed of four more chapters, three of which correspond to manuscripts that have been either published or submitted for revisions in operations research related journals. The final chapter lays out the conclusions and future research directions. Since this thesis has a manuscript-style, each chapter is self contained. The remaining of this thesis is organized as follows. In Chapter 2, we introduce a linear mixed integer programming formulation, Lagrangean relaxation and primal heuristic for the CDAP. We use the sub-gradient optimization method to solve the Lagrangean dual problem, and compare our heuristics with existing ones in the literature. New instances are introduced to further test our solution algorithms. In Chapter 3, we introduce two more formulations for the CDAP, along with linear, combinatorial and Lagrangean relaxations that are computationally and theoretically compared. One of these formulations has an exponential number of variables, and so, we introduce a column generation algorithm to obtain the optimal solution of its linear programming relaxation. In Chapter 4, we introduce a container scheduling and cross-dock door selection problem where the assignments are integrated with sequencing and selection decisions. We then introduce two integer programming formulations that cater for static and dynamic environments. New set of instances are generated to test the performance of these formulations. Finally, Chapter 5 lays out conclusions and future research directions.

Chapter 2

A Mixed-Integer Programming Formulation and Lagrangean Relaxation for the Cross-dock Door Assignment Problem

In a cross-dock, goods are unloaded from incoming trucks, consolidated according to their destinations, and then, loaded into outgoing trucks with little or no storage in between. In this paper, we address the cross-dock door assignment problem in which the assignment of incoming trucks to strip doors, and outgoing trucks to stack doors is determined, with the objective of minimizing the total material handling cost. We present a mixed integer programming formulation which is embedded into a Lagrangean relaxation that exploits the special structure of the problem to obtain bounds on the optimal solution value. A primal heuristic is used at every iteration of the Lagrangean relaxation to obtain high quality feasible solutions. Computational results obtained on benchmark instances (with up to 20 origins and destinations, and 10 strip and stack doors) and on a new and more difficult set of instances (with up to 50 origins and destinations, and 30 strip and stack doors) confirm the efficiency of the algorithm.

The following paper was published in the *International Journal of Production Research* 54.2 (2016): 494-508. Nassief et al. [42].

2.1 Introduction

A recent trend nowadays is seen in the surging number of companies adopting cross-docking techniques towards optimizing their supply chain operations. In fact, the emergence of cost-cutting, efficiency-improving philosophies such as just-in-time and lean systems has been a key factor contributing to the increasing popularity of cross-docking. In reality, cross-dock terminals are designed to facilitate rapid movements of materials, which eventually leads to a better service level and quicker response across the supply chain at a reduced cost. More specifically, once the incoming trucks arrive to their designated strip doors, the goods get unloaded, consolidated in a sorting/staging area according to their destinations, and then loaded through stack doors into the outgoing trucks with minimal storage in between. The amount of time goods spend at a cross-dock ranges from virtually none to 24 hours. This poses as one of the main differences between traditional warehouses and the more contemporary cross-dock terminals. Cross-docking, in its best practice, aims to reduce or, where possible, eliminate two major functions of a traditional warehouse: storage and order picking, assuming that relevant product and environmental settings are met. The various advantages of cross-docking as compared to the different practices of traditional warehouses and point-to-point deliveries are listed in Van Belle et al. [58].

According to Arnaout et al. [8], the US trucking industry was the first to cross-dock commodities back in the 1930s. However, it was Walmart who set the trend when it realized tremendous success in applying such strategy in the retail industry in the 1980s. The use of cross-dock terminals enabled Walmart to introduce every-day low price strategies and to virtually eliminate all inventory holding costs Chen et al. [17]. Since then, the use of cross-docks has become a standard practice among companies in several industries including the automobile industry, air transportation, less-than-truckload (LTL) carriers, and mail service and parcel delivery providers, among others. One application of cross docking can be seen in a mail distribution centre where letters and parcels are collected from various mail aggregation centres before being sorted and routed to their proper destinations. For express delivery companies, where speed is a key distinguishing business characteristic, parcel is kept on the move all the time and would typically spend a maximum of a couple of hours at their hub facility before it is sent out for delivery. Another application can be seen in an airport, where transit passengers spend typically a couple of hours since arriving at the designated gate (i.e. strip door) in a transit area before they board the outbound flight through the designated gate (i.e. stack door). The baggage is also transferred using material handling equipments from the arrival gates to a temporary internal waiting area before being routed to the different departure gates. For examples of successful cross-docking implementations, the interested reader is referred to Forger [25], Kinnear [31], Witt [60], Cook et al. [22], Chen and Song [16], and Napolitano [41].

Given the inherent complexity of designing and operating cross-docking terminals, several classes of decision problems have been studied in the literature. Van Belle et al. [58] provide a comprehensive review and classification of decision problems arising in cross-docking. Some of these problems deal with strategic decisions such as determining the optimal location of cross-docks and figuring out their best layout design. Other problems are concerned with rather tactical decisions such as determining the optimal flow of commodities through a network of cross-docks. Operational decision problems are abundant and they arise in a wide variety of contexts such as the assignment of trucks to dock doors, truck scheduling, vehicle routing for pick-up and delivery operations, and the location of products in the temporary storage area. For other reviews of decision problems arising in cross-docking, we refer to Agustina et al. [2], Boysen [10], Shuib and Fatthi [53], and Buijs et al. [14].

This paper focuses on the so-called *cross-dock door assignment problems* (CDAPs). On a daily basis, incoming trucks arriving to cross-dock terminals from several origins, each of which carrying a family of similar products, are assigned to strip doors. When docked, they are unloaded by employees who inspect and sort the products according to their destinations. Using material handling equipments such as forklifts or a system of conveyors, these products are transferred to some stack doors ready for loading on outgoing trucks that are assigned to their destinations. These forklifts keep traveling between the strip and stack doors till all products are transferred to their designated stack doors. Employees, then load these consolidated products into the outgoing trucks. Whenever an outgoing truck is filled with all needed products for its assigned destination, it leaves the terminal carrying those mixed products. CDAPs seek to optimally decide on the assignment of both incoming trucks to strip doors and outgoing trucks to stack doors so that the material handling cost inside the cross-dock is minimized. The assignment patterns affect the performance inside the terminal in terms of traveling distance by forklifts, labor's utilization, and traveling time. The cost is commonly represented as traveling distance between doors.

Peck [46] seems to be the first work addressing the assignment of inbound and outbound trucks to dock doors. The author develops a simulation model to represent the activities at an LTL terminal and uses the concept of full floating dock to allocate trucks to dock doors on a continuous basis as new shipments arrive on a short-term horizon. The considered CDAP includes capacity constraints to limit the total freight transfer time inside the LTL terminal and the objective is to minimize the total time to transfer the goods internally from inbound to outbound trucks. This problem is first formulated as a bilinear integer program and a linearized mixed integer programming (MIP) formulation is then proposed. A greedy heuristic is also presented to solve the problem. Tsui and Chang [55] study a CDAP on a mid-term horizon in which each inbound (outbound) door is assigned to only one origin (destination) point. It is assumed that the number of inbound (outbound) doors is equal to the number of origins (destinations) and the goal is to minimize the total material handling cost, stated in terms of the weighted distance traveled by forklifts. This problem is formulated as a bilinear integer program and solved by means of a simple local improvement algorithm. In a related work, Tsui and Chang [56] present a branch and bound method for this problem in which the solution of assignment problems are used to obtain lower bounds at nodes of the enumeration tree. Oh et al. [45] address a CDAP arising in a mail distribution center in which destinations have to be clustered into groups to be assigned to specific outbound doors. This problem assumes that there is only one inbound door in which all shipments are received and the objective is to minimize the travel distance of goods in the center. The authors present a nonlinear integer program to model the problem and two heuristics to solve it.

Bozer and Carlo [13] study a CDAP in which it is assumed that the number of inbound (outbound) trucks is equal to the number of inbound (outbound) doors and the objective is to minimize the total material handling cost. The authors consider two different variants of the problem: static and dynamic. In the static version, the assignment of outbound doors is assumed to be fixed while the inbound doors are assigned every night. The dynamic version, however, considers that both inbound and outbound doors have to be assigned every night. The static CDAP is solved using a simulated annealing algorithm whereas the dynamic problem is modeled as a quadratic assignment problem with rectilinear distances. Cohen and Keren [21] consider a short-term horizon to deal with the assignment of trucks to doors while considering capacity limitations on the weight of each truck. Commodities with a common destination can be shipped using more than one truck assigned to different outbound doors. This implies that a destination point can be assigned to several outbound doors. The authors model the problem as a nonlinear integer program and solve it using a heuristic algorithm. Zhu et al. [62] study a CDAP over a mid-term horizon where they extend the model of Tsui and Chang [55, 56] by considering the more realistic case in which there is a larger number of origins and destinations than that of inbound and outbound doors at a cross-dock. This model allows more than one origin (destination) to be allocated to the same inbound (outbound) door and imposes a capacity constraint on the amount of products that can be processed at each door. The problem is formulated as a nonlinear integer program and transformed into a special case of the generalized quadratic 3-dimensional assignment problem to use the branch-and-bound algorithm presented in Hahn et al. [29] to solve it. Guignard et al. [28] present two different heuristic algorithms to solve the same problem. The first approach is a local search method in which generalized assignment problems are iteratively solved to determine the best assignment of origins or destinations to dock doors. The second, however, is an adaptation of the convex hull heuristic introduced in Ahlafçioglu et al. [3]. We refer to Luo and Noble [37] and Choy et al. [20] for other CDAPs that incorporate additional features of real applications such as a limited capacity on storage and staging areas and random arrivals of inbound trucks.

Given the inherent complexity of the non-linearities that naturally arise in the formulation of CDAPs, most of the literature has resorted to heuristic algorithms for their solution. However, very little work has been done for the development of mathematical programming formulations that can be used with general purpose optimization solvers and with decomposition techniques to efficiently solve them and to provide performance guaranties on the quality of the obtained solutions. In this paper we study the CDAP considered in Zhu et al. [62] and Guignard et al. [28]. From now on, we refer to this problem as the CDAP.

We present a new linear MIP formulation for the CDAP that uses path-based variables to characterize the possible paths that commodities can use between origins and destinations passing through inbound and outbound doors. Given the large number of variables and constraints in the formulation, we propose a Lagrangean relaxation (LR) algorithm that is able to exploit the structure of the problem to obtain bounds on the optimal solution value. A primal heuristic is also developed to provide feasible solutions. In order to evaluate the efficiency and limitations of our MIP formulation and LR, computational experiments were performed on benchmark instances with up to 20 origins and destinations and 10 inbound and outbound doors.

The results show that our MIP formulation can optimally solve most of the considered instances when used with a general purpose solver. In particular, it is able to provide the optimal solution to five benchmark instances for which the optimal solution was not known before. Moreover, the LR algorithm is capable of providing better solutions on average than the ones obtained with the heuristics of Guignard et al. [28] in small CPU times with a performance guarantee. Additional experiments were also performed on a new challenging set of instances with up to 50 origins and destinations and 30 inbound and outbound doors. The LR is able to obtain on average better optimality gaps than CPLEX in two hours of CPU time, specially for the largest-size instances with 50 origins and destinations.

The remainder of this paper is structured as follows. Section 2 provides a formal definition of the problem along with the proposed mathematical programming formulation. In Sections 3 and 4 we describe the LR algorithm and the primal heuristic, respectively. The data generation technique and the computational experiments for existing benchmark instances and for the newly developed set of instances are presented in Section 5. Conclusions follow in Section 6.

2.2 Definition and Formulations of the Problem

Let M, N, I, and J denote the set of origins, destinations, inbound doors and outbound doors, respectively. Let d_{ij} denote the distance (or travel time) between inbound door i and outbound door j. Let w_{mn} be the amount of commodity originated at m with destination n. The values w_{mn} can be standardized so as to be stated in terms of the number of times the material handling equipment (i.e. forklift, pallet jack, etc) needs to be used to transfer all the flow originated at m with destination n between inbound and outbound doors. Let $s_m = \sum_{n \in N} w_{mn}$ denote the total flow going out of an origin while $r_n = \sum_{m \in M} w_{mn}$ the total flow coming into a destination. We denote as S_i and R_j the capacity for inbound door $i \in I$ and outbound door $j \in J$, respectively. This capacity denotes the limit on the amount of commodities the material handling equipment designated to each door can process in the considered period of time. For each origin/destination pair, the material handling cost when commodity w_{mn} is received at inbound door i and shipped from outbound door j is given by $w_{mn}d_{ij}$.

The CDAP consists of assigning each origin and each destination to exactly one inbound door and one outbound door, respectively, in such a way that the capacity constraints of dock doors are satisfied and the total material handling cost inside the cross-dock is minimum.

2.2.1 A Nonlinear Integer Programming Formulation

A natural way to formulate the CDAP is to use two sets of binary decision variables to determine the assignment pattern of origins and destinations to dock doors. For each pair $m \in M$, $i \in I$, we define

$$x_{mi} = \begin{cases} 1 & \text{if origin } m \text{ is assigned to inbound door } i; \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, for each pair $n \in N$, $j \in J$, we define

$$y_{nj} = \begin{cases} 1 & \text{if destination } n \text{ is assigned to outbound door } j; \\ 0 & \text{otherwise.} \end{cases}$$

Using these sets of variables, Zhu et al. [62] formulate the CDAP as the following bilinear integer program:

$$\begin{aligned} (P1) \quad \text{minimize} \quad & \sum_{m\in M}\sum_{i\in I}\sum_{j\in J}\sum_{n\in N} w_{mn} d_{ij} x_{mi} y_{nj} && && (2.1) \\ \text{subject to} \quad & \sum_{i\in I} x_{mi} = 1 && m \in M && (2.2) \\ & \sum_{j\in J} y_{nj} = 1 && n \in N && (2.3) \\ & \sum_{m\in M} s_m x_{mi} \le S_i && i \in I && (2.4) \\ & \sum_{n\in N} r_n y_{nj} \le R_j && j \in J && (2.5) \\ & x_{mi} \in \{0,1\} && m \in M, i \in I && (2.6) \\ & y_{nj} \in \{0,1\} && n \in N, j \in J. && (2.7) \end{aligned}$$

The objective function seeks to minimize the total weighted distance traveled by the material handling equipment. Constraints (2.2) and (2.3) guarantee that every origin (destination) is assigned to exactly one inbound (outbound) door. Constraints (2.4) and (2.5) ensure that the capacity constraints on the inbound and outbound doors, respectively, are satisfied. Constraints (2.6) and

(2.7) are the classical integrality conditions on the decision variables. Note that constraints (2.2)-(2.7) define the set of feasible solutions of two independent *generalized assignment problems* and the quadratic term of the objective (2.1) links them.

2.2.2 A Linear Mixed Integer Programming Formulation

We next provide a new linear MIP formulation that exploits the structure of the problem and that can be efficiently adapted to use in decomposition techniques. Let K represent the set of commodities whose origin and destination points belong to M and N, respectively. For each commodity $k \in K$, define as w_k the amount of commodity k to be routed from the origin $o(k) \in M$ to the destination $d(k) \in N$. We use the following set of binary decision variables to characterize the possible paths that commodities can follow at the cross-dock. For each $k \in K$, $i \in I$, $j \in J$ let

$$z_{kij} = \begin{cases} 1 & \text{if commodity } k \text{ transits via inbound door } i \text{ and outbound door } j; \\ 0 & \text{otherwise.} \end{cases}$$

Using this set of variables and the x_{mi} and y_{nj} variables previously defined, the CDAP can be modeled as the following linear MIP program:

$$\begin{aligned}
(P2) \quad \text{minimize} \quad & \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} w_k d_{ij} z_{kij} & & \\
\text{subject to} \quad & \sum_{i \in I} \sum_{j \in J} z_{kij} = 1 & k \in K & \quad (2.8) \\
& \sum_{j \in J} z_{kij} = x_{o(k)i} & k \in K, i \in I & \quad (2.9) \\
& \sum_{i \in I} z_{kij} = y_{d(k)j} & k \in K, j \in J & \quad (2.10) \\
& \sum_{m \in M} s_m x_{mi} \leq S_i & i \in I & \quad (2.11) \\
& \sum_{n \in N} r_n y_{nj} \leq R_j & j \in J & \quad (2.12) \\
& x_{mi} \in \{0,1\} & m \in M, i \in I & \quad (2.13) \\
& y_{nj} \in \{0,1\} & n \in N, j \in J & \quad (2.14) \\
& z_{kij} \geq 0 & k \in K, i \in I, j \in J. & \quad (2.15)
\end{aligned}$$

Constraints (2.8) ensures that there must be a single path connecting the origin and destination nodes of each commodity with a pair of inbound and outbound doors. Constraints (2.9) state that if the origin of commodity k is assigned to inbound door i, then it must be routed through inbound door i and some outbound door j. In a similar fashion, (2.10) state that if the destination of commodity k is assigned to an outbound door j then it must be routed through outbound door j and some inbound door i. Constraints (2.11) and (2.12) make sure the capacity of inbound and outbound doors are not exceeded, respectively. Finally, constraints (2.13)-(2.15) are the standard integrality and nonnegativity conditions on the decision variables.

Note that in formulation P2 the assignment constraints (2.2)-(2.3) are not required to guarantee the single allocation pattern of origins and destinations to inbound and outbound doors, respectively. Constraints (2.8) and (2.9), together with the integrality conditions of the x_{mi} variables, ensure that all commodities originated at m are routed via a single inbound door i. Similarly, constraints (2.8) and (2.10), together with the integrality conditions of the y_{nj} variables, guarantee that all commodities with destination n are routed via a single outbound door j. Moreover, there is no need to explicitly state the integrality conditions on the z_{kij} variables given that equations (2.8)-(2.10) have no feasible fractional solutions whenever the x_{mi} and y_{nj} variables take a binary value.

Capacity constraints (2.11) use assignment variables x_{mi} to directly compute the total incoming flow at inbound door i coming from a set of origin nodes to ensure the capacity S_i is not violated. The z_{kij} and y_{nj} variables can also be used to indirectly impose the capacity constraints at inbound door i. In particular, for each door pair $(i, j) \in I \times J$ and destination node $n \in N$, we use the z_{kij} variables to compute the amount of flow with destination $n \in N$ entering to inbound door i coming from any origin $m \in M$ as

$$\sum_{k \in K: d(k)=n} w_k z_{kij} \leq S_i y_{nj} \qquad n \in N, j \in J, i \in I.$$

Contrary to (2.11), these constraints have the capacities S_i being multiplied by the y_{nj} variables, which can cause the lower bounds associated with the linear programming (LP) relaxation of P2 to improve. These constraints can be further strengthened by lifting the coefficients of the z_{kij} variables using the fact that origins nodes can only be assigned to one inbound door. That is, if a commodity originated at node m with destination n is being routed through inbound door i, then all the commodities originated at the same origin have to be routed trough the same inbound door regardless of their destination. Therefore, the following inequalities are valid for the set of feasible integer solutions of P2:

$$\sum_{k \in K: d(k)=n} s_{o(k)} z_{kij} \leq S_i y_{nj} \qquad n \in N, j \in J, i \in I. \tag{2.16}$$

Similarly, we can use the z_{kij} and x_{mi} variables to indirectly impose the capacity constraints at outbound doors. In particular, for each door pair $(i, j) \in I \times J$ and origin node $m \in M$, we use the z_{kij} variables to compute the amount of flow with origin $m \in M$ entering to outbound door j and having as destination any node $n \in N$ as

$$\sum_{k \in K: o(k)=m} r_{d(k)} z_{kij} \leq R_j x_{mi} \qquad m \in M, i \in I, j \in J. \tag{2.17}$$

Note that constraints (2.16)-(2.17) are redundant to formulation P2. However, as we show in Section 2.5, these constraints can improve the LP bounds of P2 at the expense of considerably increasing the number of constraints. We denote by P3 the extended MIP formulation obtained by adding (2.16)-(2.17) into P2. We next show how we can use a decomposition method to exploit the structure of formulation P3 to efficiently obtain lower and upper bounds on the optimal solution value.

2.3 Lagrangean Relaxation

LR is a well-known method for solving large-scale combinatorial optimization problems Geoffrion [27]. It exploits the inherent structure of the problems to compute lower bounds on the value of the optimal solution. In the case of formulation P3, if we relax the constraints (2.9)-(2.10) together with (2.16)-(2.17) in a Lagrangean fashion, weighting their violations with a multiplier vector $(\mu, \nu, \lambda, \gamma)$ of appropriate dimension, we obtain the following Lagrangean function:

$$\begin{aligned}
L(\mu, \nu, \lambda, \gamma) = \text{minimize} \quad & \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} w_k d_{ij} z_{kij} \\
& + \sum_{k \in K} \sum_{i \in I} \mu_{ki} \left(\sum_{j \in J} z_{kij} - x_{o(k)i} \right) \\
& + \sum_{k \in K} \sum_{j \in J} \nu_{kj} \left(\sum_{i \in I} z_{kij} - y_{d(k)j} \right) \\
& + \sum_{m \in M} \sum_{i \in I} \sum_{j \in J} \lambda_{mij} \left(\sum_{k \in K : o(k) = m} r_{d(k)} z_{kij} - R_j x_{mi} \right) \\
& + \sum_{n \in N} \sum_{j \in J} \sum_{i \in I} \gamma_{nji} \left(\sum_{k \in K : d(k) = n} s_{o(k)} z_{kij} - S_i y_{nj} \right) \\
\text{subject to} \quad & (2.8), (2.11) - (2.15).
\end{aligned}$$

Note that $L(\mu, \nu, \lambda, \gamma)$ can be decomposed into three subproblems: (1) a problem in the space of z variables, (2) a problem in the space of x variables, and (3) a problem in the space of y variables. After some algebra, the first subproblem can be expressed as

$$L_z(\mu, \nu, \lambda, \gamma) = \text{minimize} \sum_{k \in K} \sum_{i \in I} \sum_{j \in J} Q_{kij} z_{kij}$$

$$\text{subject to (2.8) and (2.15),}$$

where $Q_{kij} = \left(w_k d_{ij} + \mu_{ki} + \nu_{kj} + r_{d(k)} \lambda_{o(k)ij} + s_{o(k)} \gamma_{d(k)ji} \right)$. The second subproblem can be expressed as

$$L_x(\mu, \lambda) = \text{maximize} \sum_{m \in M} \sum_{i \in I} \left(\sum_{k \in K : o(k) = m} \mu_{ki} + \sum_{j \in J} R_j \lambda_{mij} \right) x_{mi}$$

$$\text{subject to (2.11) and (2.13),}$$

and the third subproblem can be expressed as

$$L_y(\nu, \gamma) = \text{maximize} \sum_{n \in N} \sum_{j \in J} \left(\sum_{k \in K : d(k) = n} \nu_{kj} + \sum_{i \in I} S_i \gamma_{nji} \right) y_{nj}$$

$$\text{subject to (2.12) and (2.14).}$$

Therefore, we have:

$$L(\mu, \nu, \lambda, \gamma) = L_z(\mu, \nu, \lambda, \gamma) + L_x(\mu, \lambda) + L_y(\nu, \gamma).$$

2.3.1 Solution to Subproblem $L_z(\mu, \nu, \lambda, \gamma)$

Subproblem $L_z(\mu, \nu, \lambda, \gamma)$ is a semi-assignment problem that, in turn, can be decomposed into $|K|$ independent semi-assignment problems, corresponding to each commodity $k \in K$, of the form

$$\begin{aligned} (SAP_k) \text{ minimize } & \sum_{i \in I} \sum_{j \in J} Q_{kij} z_{kij} \\ \text{subject to } & \sum_{i \in I} \sum_{j \in J} z_{kij} = 1 \\ & z_{kij} \geq 0 \qquad i \in I, j \in J. \end{aligned}$$

Each SAP_k problem can be easily solved by choosing exactly one door pair, denoted as $(i(k), j(k)) \in I \times J$, among those with minimum material handling cost. For a given k, an optimal solution of SAP_k, denoted by $z(\mu, \nu, \lambda, \gamma)$, can be obtained by setting exactly one z_{kij} variable to one and the rest to zero, i.e., $z_{ki(k)j(k)} = 1$ for one element $(i(k), j(k)) \in \arg\min \{Q_{kij} : (i, j) \in I \times J\}$ and $z_{kij} = 0$, for $(i, j) \in I \times J \setminus \{(i(k), j(k))\}$. The optimal solution value of SAP_k can thus be expressed as

$$L_z(\mu, \nu, \lambda, \gamma) = \sum_{k \in K} Q_{ki(k)j(k)} = \sum_{k \in K} \min \{Q_{kij} : (i, j) \in I \times J\}.$$

2.3.2 Solution to Subproblems $L_x(\mu, \lambda)$ and $L_y(\nu, \gamma)$

Subproblems $L_x(\mu, \lambda)$ and $L_y(\nu, \gamma)$ can be further decomposed into $|I|$ and $|J|$ independent binary knapsack problems, respectively. For each inbound door $i \in I$, we have

$$\begin{aligned} (KP_i) = \text{maximize } & \sum_{m \in M} \left(\sum_{k \in K : o(k) = m} \mu_{ki} + \sum_{j \in J} R_j \lambda_{mij} \right) x_{mi} \\ \text{subject to } & \sum_{m \in M} s_m x_{mi} \leq S_i && (2.18) \\ & x_{mi} \in \{0, 1\} \qquad m \in M, && (2.19) \end{aligned}$$

and for each outbound door $j \in J$, we have

$$(KP_j) = \text{maximize} \sum_{n \in N} \left(\sum_{k \in K : d(k)=n} \nu_{kj} + \sum_{i \in I} S_i \gamma_{nji} \right) y_{nj}$$

$$\text{subject to} \sum_{n \in N} r_n y_{nj} \leq R_j \tag{2.20}$$

$$y_{nj} \in \{0, 1\} \qquad n \in N. \tag{2.21}$$

Each KP_i and KP_j problem evaluates the benefit of assigning one or more origins to an inbound door and one or more destinations to an outbound door, respectively. Although knapsack problems are known to belong to the class of NP-hard problems, they can be solved efficiently using the algorithm of Martello et al. [39].

2.3.3 The Solution of the Lagrangean Dual

In order to obtain the best lower bound, one must solve the following Lagrangean dual problem

$$(D) \quad z_D = \max_{\lambda, \gamma \geq 0} L(\mu, \nu, \lambda, \gamma)$$

We apply the subgradient optimization method to solve problem D. Now, for a given vector $(\mu, \nu, \lambda, \gamma)$, let $z(\mu, \nu, \lambda, \gamma)$, $x(\mu, \lambda)$ and $y(\nu, \gamma)$ denote the optimal solution to $L(\mu, \nu, \lambda, \gamma)$. Then, a subgradient of $L(\mu, \nu, \lambda, \gamma)$ is given by:

$$\delta(\mu, \nu, \lambda, \gamma) = \Bigg(\left(\sum_{j \in J} z_{kij}(\mu, \nu, \lambda, \gamma) - x_{o(k)i}(\mu, \lambda) \right)_{ki}, \left(\sum_{i \in I} z_{kij}(\mu, \nu, \lambda, \gamma) - y_{d(k)j}(\nu, \gamma) \right)_{kj},$$
$$\left(\sum_{\substack{k \in K \\ o(k)=m}} r_{d(k)} z_{kij}(\mu, \nu, \lambda, \gamma) - R_j x_{mi}(\mu, \lambda) \right)_{mij},$$
$$\left(\sum_{\substack{k \in K \\ d(k)=n}} s_{o(k)} z_{kij}(\mu, \nu, \lambda, \gamma) - S_i y_{nj}(\nu, \gamma) \right)_{nji} \Bigg).$$

We use the following standard formula to update the dual multipliers at every iteration t of the subgradient algorithm:

$$(\mu, \nu, \lambda, \gamma)^{t+1} = (\mu, \nu, \lambda, \gamma)^t + \epsilon^t \frac{(\overline{\eta} - L\left((\mu, \nu, \lambda, \gamma)^t\right))}{\|\delta((\mu, \nu, \lambda, \gamma)^t)\|^2} \delta((\mu, \nu, \lambda, \gamma)^t),$$

where $\bar{\eta}$ denotes an upper bound on the optimal value of P3 and $0 < \epsilon^t < 2$ is a parameter to control the step length used at every iteration.

A scheme of the subgradient algorithm is depicted in Algorithm 1. The output of the algorithm is a lower bound z_D.

Algorithm 1 Subgradient Method

```
Iteration 0
  Initialize $z_D \leftarrow -\infty$ ; $(\mu, \nu, \lambda, \gamma)^0 \leftarrow 0$ ; $\epsilon^0 \leftarrow 2$.
  Let $\bar{\eta}$ be a known upper bound on the optimal solution value
Iteration $t$
  Solve the Lagrangean function $L((\mu, \nu, \lambda, \gamma)^t)$
  if $(L((\mu, \nu, \lambda, \gamma)^t) > z_D)$ then
    $z_D \leftarrow L((\mu, \nu, \lambda, \gamma)^t)$
  end if
  Evaluate the subgradient $\delta((\mu, \nu, \lambda, \gamma)^t)$
  Calculate the step length $\phi^t \leftarrow \epsilon^t \frac{\left(\bar{\eta} - L((\mu,\nu,\lambda,\gamma)^t)\right)}{\|\delta((\mu,\nu,\lambda,\gamma)^t)\|^2}$
  $(\mu, \nu, \lambda, \gamma)^{t+1} \leftarrow (\mu, \nu, \lambda, \gamma)^t + \phi^t \delta((\mu, \nu, \lambda, \gamma)^t)$
  $t \leftarrow t + 1$
```

2.4 Primal Heuristic

We next propose a primal heuristic that is applied at every iteration of the subgradient optimization algorithm to obtain feasible solutions for the CDAP. It uses information generated by the evaluation of the Lagrangean function to obtain an initial feasible solution, which is later improved by a local search procedure. In what follows, solutions are represented by pairs of the form $s = (a, b)$, where $a : M \rightarrow I$ and $b : N \rightarrow J$ denote the assignment mapping of origins and destinations to inbound and outbound doors, respectively. That is, $a(m) = i$ if origin m is assigned to inbound door i and $b(n) = j$ if destination n is assigned to outbound door j. For any assignment, h_i and q_j denote the available capacity at inbound door i and outbound door j, respectively. That is, $h_i = S_i - \sum_{m:a(m)=i} s_m$ and $q_j = R_j - \sum_{n:b(n)=j} r_n$. From now on, we drop the reference to the vector of dual multipliers $(\mu, \nu, \lambda, \gamma)$ and thus, we write L_x, L_y, x_{mi}, and y_{nj} instead of $L_x(\mu, \lambda)$,

$L_y(\nu, \gamma)$, $x_{mi}(\mu, \lambda)$, and $y_{nj}(\nu, \gamma)$.

2.4.1 Constructive Phase

The solution to each KP_i and KP_j subproblem provides a feasible assignment with respect to the capacity constraints of inbound and outbound doors. However, these problems are solved independently and thus, solutions may be infeasible for the CDAP for two reasons. First, an origin (destination) can be assigned to more than one inbound (outbound) door. Second, an origin (destination) may not be assigned to any inbound (outbound) door. Therefore, exactly one of the following three scenarios occurs for each origin and each destination in any solution of L_x and L_y: *i)* it is assigned to exactly one door, *ii)* it is assigned to more than one door, and *iii)* it is not assigned to any door.

We construct an initial solution as follows. We first consider the sets of origins and destinations assigned to exactly one door, i.e.

$$AS_1^M = \left\{ m \in M : \sum_{i \in I} x_{mi} = 1 \right\} \text{ and } AS_1^N = \left\{ n \in N : \sum_{j \in J} y_{nj} = 1 \right\},$$

and fix these assignments. That is, $a(m) = i(m)$ for every $m \in AS_1^M$, where $a_{mi(m)} = 1$, and $b(n) = j(n)$ for every $n \in AS_1^N$, where $y_{nj(n)} = 1$. We then consider the sets of origins and destinations assigned to more than one door, i.e.

$$AS_2^M = \left\{ m \in M : \sum_{i \in I} x_{mi} > 1 \right\} \text{ and } AS_2^N = \left\{ n \in N : \sum_{j \in J} y_{nj} > 1 \right\},$$

and assign each one to an arbitrary door that was allocated to that satisfies the capacity constraints. That is, when feasible we set $a(m) = i^*$ for $m \in AS_2^M$, where $i^* \in \{i : x_{mi} = 1, h_i - s_m \geq 0\}$ and $b(n) = j^*$ for $n \in AS_2^N$, where $j^* \in \{j : y_{nj} = 1, q_j - r_n \geq 0\}$. Note that all origins and destinations in AS_2^M and AS_2^N respect the capacity constraints as a result of the knapsack solution. Finally, we order the remaining unassigned origins and destinations in a decreasing order with respect to their total outgoing and incoming flow s_m and r_n, respectively, and assign them, one at a time, to an arbitrarily door that satisfies its capacity constraint after the assignment is made. Whenever we obtain an initial feasible solution with this constructive procedure, we try to improve it with the following local search procedure.

2.4.2 Local Search Phase

The proposed local search procedure uses two classes of neighborhoods to improve the initial solution obtained with the constructive phase. The first class modifies the assignments of either origins or destinations independently by using the classical *swap* and *shift* neighborhoods. The *shift* neighborhood considers all solutions that can be reached from the current one by changing the assignment of exactly one origin (or destination), whereas the *swap* neighborhood considers all solutions that differ from the current one in the assignment of two origins (or destinations). Let $s = (a, b)$ be the current solution, then the shift neighborhood associated with the origins is defined as

$$N^{M}_{shift}(s) = \left\{ s^{'} = (a', b) : \exists ! m \in M, a'(m) \neq a(m) \right\},$$

and the shift neighborhood associated with the destinations as

$$N^{N}_{shift}(s) = \left\{ s^{'} = (a, b') : \exists ! n \in N, b'(n) \neq b(n) \right\}.$$

For exploring $N^{M}_{shift}(s)$ we consider all pairs $(m, i) \in M \times I$, such that $h_i - s_m \geq 0$ and, similarly, for $N^{N}_{shift}(s)$ we consider all pairs $(n, j) \in N \times J$, such that $q_j - r_n \geq 0$. The swap neighborhood associated with the origins is defined as

$$\begin{aligned} N^{M}_{swap}(s) = \Big\{ & s^{'} = (a', b) : \exists m_1, m_2 \in M, a'(m_1) = a(m_2), a'(m_2) = a(m_1), \\ & a'(m) = a(m) \forall m \neq m_1, m_2 \right\}, \end{aligned}$$

and the swap neighborhood associated with the destinations as

$$\begin{aligned} N^{N}_{swap}(s) = \Big\{ & s^{'} = (a, b') : \exists n_1, n_2 \in N, b'(n_1) = b(n_2), b'(n_2) = b(n_1), \\ & b'(n) = b(n), \forall n \neq n_1, n_2 \right\}. \end{aligned}$$

For exploring $N^{M}_{swap}(s)$ we consider all pairs $(m_1, m_2) \in M \times M$, such that $a(m_1) \neq a(m_2)$, $h_{a(m_1)} + s_{m_1} - s_{m_2} \geq 0$ and $h_{a(m_2)} + s_{m_2} - s_{m_1} \geq 0$. Similarly, for exploring $N^{N}_{swap}(s)$ we consider all pairs $(n_1, n_2) \in N \times N$, such that $b(n_1) \neq b(n_2)$, $q_{b(n_1)} + r_{n_1} - r_{n_2} \geq 0$ and $q_{b(n_2)} + r_{n_2} - r_{n_1} \geq 0$.

The second class of neighborhoods modifies the assignment of origins and destinations at the same time. In particular, the *double shift* neighborhood considers all solutions that can be reached from the current one by changing the assignment of exactly one origin and one destination at the same time and is defined as

$$N_{shift}^{MN} = \left\{ s' = (a', b') : \exists!(m,n) \in M \times N, a'(m) \neq a(m), b'(n) \neq b(n) \right\}.$$

For exploring N_{shift}^{MN} we consider all $(m,i) \in M \times I$ and $(n,j) \in N \times J$, such that $h_i - s_m \geq 0$ and $q_j - r_n \geq 0$. In the first four neighborhoods we use a best improvement strategy while in the last one we use a first improvement strategy.

Algorithm 2 describes a summary of the proposed local search procedure.

Algorithm 2 Local Search

```
while solution is improved do
    Explore $N_{shift}^{M}(s) \cup N_{shift}^{N}(s) \cup N_{swap}^{M}(s) \cup N_{swap}^{N}(s)$
    if no improvement then
        Explore $N_{shift}^{MN}$
    end if
end while
```

2.5 Computational Experiments

We next present the results of extensive computational experiments performed to assess the behavior of the MIP formulations and Lagrangean relaxation for the CDAP. All algorithms were coded in C and run on an Intel Xeon E3 1240 V2 processor with 3.40 GHz and 24GB of RAM memory under a Windows environment. The MIP formulation was implemented using the callable library of CPLEX 12.5.2. The knapsack problems were solved with the exact algorithm described in Martello et al. [39]. In all the LR experiments, the subgradient optimization algorithm terminates when one of the following criteria is met:

i) The difference between the upper and lower bounds is below a threshold value, i.e. $z^* - z_D^t \leq \chi$;

ii) The maximum number of iterations Iter$_{max}$ is reached;

iii) The CPU time limit Time$_{max}$ is reached.

After some tuning, we set the following parameters values: $\chi = 10^{-7}$, Iter$_{max} = 30,000$, and Time$_{max} = 7,200$ seconds. Moreover, the parameter ϵ^t is multiplied by 0.1 every 5,000 consecutive iterations without improvement in the lower bound and is reset to 2 whenever $\epsilon^t < 10^{-7}$.

We have used two sets of benchmark instances in our computational experiments: the data set introduced by Guignard et al. [28] and a new challenging set of instances generated by us. In Guignard et al. [28], the authors generate the flow matrix with integer numbers between $[10, 50]$ until 25% of the matrix is full. They ensure that each destination receives some flow from at least one origin and each origin sends some flow to at least one destination. The distance matrix is generated with numbers from the interval $[8, 8 + I - 1]$ indicating that a direct distance between two doors (i.e. two doors facing each other) is 8, and then an increment of 1 unit is added for the next indirect door. All instances are generated with $|I| = |J|$ for a rectangular shaped cross-dock. Lastly, capacities are identical and calculated by dividing the total flow coming from all origins by the total number of inbound doors, and then adding a capacity slackness of 5%, 10%, 15%, %20 and %30. The number of considered origins/destinations is $\{8, 9, 10, 11, 12, 15, 20\}$ and the number of considered inbound/outbound doors is $\{4, 5, 6, 7, 10\}$. A total of 50 instances are considered from the possible combinations of these parameters and are denoted as 00x00S00, where the first position refers to the number of origins/destinations, the second position to the number of inbound/outbound doors, and the last position to the capacity slackness. That is, 20x10S15 denotes an instance with 20 origins, 20 destinations, 10 inbound doors, 10 outbound doors, and a capacity slackness of 15%.

In the first part of the experiments, we compare the results obtained with formulations P2 and P3 when used with a general purpose solver, our Lagrangean relaxation algorithm, and the heuristics presented in Guignard et al. [28]. The detailed results of this comparison using the first set of benchmark instances are given in Tables 2.1 and 2.2. The first column gives the name of the instance. The next six columns under the heading *CPLEX* provide the following results: *i)* the duality gap %LP$_{P2}$ and % LP$_{P3}$ relative to the LP relaxation bound obtained with formulations P2 and P3, respectively, *ii)* the remaining optimality gap, denoted as %gap, after CPU time limit with formulation P2, *iii)* the deviation between the best solution obtained by CPLEX with formulation P2 and the optimal (or best known) solution, denoted as %dev, and *iv)* the number of branch and bound nodes and the CPU time to reach the optimal solution with formulation P2. The LP deviation is calculated as $\%LP = 100(OPT - LP)/(OPT)$, where OPT is the optimal value and LP is the optimal solution for the LP relaxation of each formulation. The deviation is calculated as $\%dev = 100(UB - OPT)/(OPT)$, where UB is the best upper bound obtained by CPLEX and OPT is the optimal (or best known) solution obtained by any method. Preliminary computational experiments showed that, in all considered instances, the required CPU time to solve formulation P3 with CPLEX was considerably larger than the time required to solve formulation P2. Therefore, we do not include the results associated with the optimal solution of CDAP with formulation P3 in this or subsequent tables. The next three columns under the heading *Lagrangean relaxation* provide the following results: *i)* the duality gap, denoted as %gap, relative to the LR bound and the best upper bound obtained by the primal heuristic, *ii)* the deviation %dev between the best solution obtained by the primal heuristic and the best known solution, and *iii)* the CPU time in seconds for the overall LR algorithm. For comparative purposes, the last three columns show the deviation between the solutions obtained with three different heuristics presented in Guignard et al. [28] and the best known solutions. We set the CPU time limit to 7,200 seconds (2 hours). Whenever CPLEX is unable to solve an instance within the CPU time limit, we indicate *time* in the corresponding entry of the table.

The results presented in Tables 2.1 and 2.2 appear to be promising. As can be seen, formulation P2 used with CPLEX is able to prove the optimality of the obtained solutions in 45 out of 50 considered instances. The results given in Guignard et al. [28] show that the branch-and-bound algorithm described in Hahn et al. [29] can only solve 40 instances. That is, formulation P2 proves optimality for five instances for which the optimal solution was not known. The columns under the heading %LP$_{P2}$ and % LP$_{P3}$ indicate that constraints (2.16)-(2.17), although redundant to formulation P2, have a positive impact on improving the LP bounds. For the considered instances, the average optimality gap is 8.54% and 5.30% for formulations P2 and P3, respectively. However, preliminary computational experiments showed that the improvement of the LP bounds does not compensate the increase of the CPU time required to solve much larger LPs at the nodes of the enumeration tree.

In terms of our LR algorithm, it is able to obtain an average optimality gap of 6.39%, which is better than formulation P2 and slightly worse than P3. This is partially explained by the slow convergence of the subgradient method. Preliminary experiments showed that when the maximum number of iterations is increased, the LR can slightly improve the lower bounds at the expense of

Table 2.1: Comparison of Results for P2 and P3 Formulations and Lagrangean Relaxation.

	CPLEX						Lagrangean Relaxation			Heuristics		
Instance	% LP$_{P2}$	% LP$_{P3}$	% gap	% dev	nodes	time	% LR	% dev	time	% LS1	% LS2	% CHR
8x4S5	6.76	3.26	0.00	0.00	256	0.34	3.51	0.00	4.62	0.00	0.00	0.00
8x4S10	6.67	3.39	0.00	0.00	222	0.29	3.63	0.00	5.19	0.00	0.00	0.00
8x4S15	5.63	2.53	0.00	0.00	235	0.26	2.87	0.00	5.38	0.00	0.00	0.00
8x4S20	5.15	2.24	0.00	0.00	97	0.47	2.49	0.00	5.47	0.00	0.00	0.00
8x4S30	4.72	2.16	0.00	0.00	170	0.45	3.62	0.00	7.12	0.00	0.00	0.00
9x4S5	7.66	3.71	0.00	0.00	255	0.41	4.19	0.00	6.45	0.00	0.00	0.00
9x4S10	7.35	3.69	0.00	0.00	197	0.56	4.17	0.00	6.16	0.00	0.00	0.00
9x4S15	6.56	3.16	0.00	0.00	171	0.30	3.54	0.00	14.04	0.00	0.00	0.00
9x4S20	5.95	2.75	0.00	0.00	115	0.38	3.34	0.00	14.69	0.00	0.00	0.00
9x4S30	5.42	2.63	0.00	0.00	247	0.40	4.41	0.00	11.32	0.00	0.00	0.00
10x4S5	8.56	6.31	0.00	0.00	1432	0.63	6.69	0.00	9.90	0.00	0.00	0.00
10x4S10	5.77	3.67	0.00	0.00	368	0.53	4.62	0.00	10.16	0.00	0.00	0.00
10x4S15	5.34	3.45	0.00	0.00	309	0.61	4.56	0.00	12.17	0.00	0.00	0.00
10x4S20	4.90	3.19	0.00	0.00	353	0.50	4.49	0.00	12.61	0.00	0.00	0.00
10x4S30	3.76	2.35	0.00	0.00	57	0.47	3.66	0.00	13.40	0.00	0.00	0.00
10x5S5	9.79	6.48	0.00	0.00	5409	1.11	7.20	0.00	9.24	0.00	0.00	0.00
10x5S10	7.84	4.78	0.00	0.00	2796	1.27	5.71	0.00	11.96	0.00	0.00	0.00
10x5S15	6.71	3.87	0.00	0.00	1090	0.88	5.18	0.00	12.18	0.00	0.00	0.00
10x5S20	5.90	3.31	0.00	0.00	803	0.63	4.56	0.00	12.86	0.50	0.33	0.19
10x5S30	5.39	3.17	0.00	0.00	729	0.76	4.32	0.00	12.69	0.25	0.40	0.40
11x5S5	10.80	7.33	0.00	0.00	48676	9.08	8.36	0.00	14.91	0.00	0.00	0.00
11x5S10	7.98	4.77	0.00	0.00	3661	1.95	6.02	0.00	16.51	0.00	0.00	0.00
11x5S15	7.52	4.59	0.00	0.00	2068	1.79	5.97	0.00	14.09	0.09	0.09	0.09
11x5S20	6.33	3.61	0.00	0.00	246	0.83	5.00	0.00	17.60	0.35	0.35	0.52
11x5S30	6.09	3.76	0.00	0.00	1863	1.53	5.32	0.00	19.48	0.11	0.05	0.00

Table 2.2: Comparison of Results for P2 and P3 Formulations and Lagrangean Relaxation.

	CPLEX						Lagrangean Relaxation			Heuristics		
Instance	% LP$_{P2}$	% LP$_{P3}$	% gap	% dev	nodes	time	% LR	% dev	time	% LS1	% LS2	% CHR
12x5S5	8.23	5.28	0.00	0.00	3774	2.52	6.06	0.00	12.24	0.00	0.00	0.00
12x5S10	7.14	4.45	0.00	0.00	1160	0.96	5.58	0.00	15.62	0.00	0.13	0.16
12x5S15	6.69	4.21	0.00	0.00	521	0.86	5.44	0.00	15.31	0.00	0.00	0.00
12x5S20	6.69	4.46	0.00	0.00	2329	1.31	5.77	0.00	18.90	0.00	0.00	0.00
12x5S30	6.50	4.61	0.00	0.00	6620	5.46	6.29	0.00	17.26	0.24	0.03	0.03
12x6S10	9.41	6.06	0.00	0.00	4332	3.41	6.57	0.00	17.72	0.18	0.38	0.23
12x6S15	8.59	5.42	0.00	0.00	3352	2.28	6.36	0.00	19.65	0.32	0.56	0.12
12x6S20	8.15	5.19	0.00	0.00	6727	3.27	6.33	0.00	23.16	0.18	0.26	0.11
12x6S30	7.39	4.82	0.00	0.00	12086	5.03	6.51	0.00	29.26	0.00	0.00	0.48
15x6S5	9.93	6.14	0.00	0.00	30450	17.56	7.05	0.01	30.87	0.24	0.00	0.40
15x6S10	9.12	5.65	0.00	0.00	14573	10.84	6.66	0.00	32.87	0.38	0.35	0.50
15x6S15	8.87	5.66	0.00	0.00	26773	18.64	6.97	0.00	35.63	0.03	0.25	0.00
15x6S20	8.57	5.61	0.00	0.00	53847	30.33	7.19	0.00	29.57	0.36	0.25	0.00
15x6S30	7.54	4.98	0.00	0.00	13430	8.43	6.77	0.00	31.36	0.00	0.00	0.43
15x7S5	13.70	8.51	0.00	0.00	226374	195.87	9.97	0.51	21.54	0.00	0.00	0.28
15x7S10	12.28	7.44	0.00	0.00	88481	83.08	8.64	0.03	24.28	0.21	0.05	0.00
15x7S15	11.36	6.80	0.00	0.00	53092	46.47	8.13	0.00	25.45	0.14	0.16	0.14
15x7S20	10.49	6.19	0.00	0.00	22803	21.22	7.81	0.00	34.73	0.56	0.13	0.13
15x7S30	9.83	6.06	0.00	0.00	55572	47.58	7.90	0.01	42.17	0.04	0.10	0.03
20x10S5	17.98	12.16	14.99	0.25	373813	time	13.40	0.51	63.76	0.04	0.33	0.00
20x10S10	16.16	10.67	11.82	0.18	231224	time	11.72	0.15	69.21	0.00	0.00	0.72
20x10S15	15.73	10.61	12.74	0.00	529189	time	11.72	0.67	79.04	0.49	0.17	1.35
20x10S20	15.23	10.47	9.68	0.00	461003	time	11.82	0.00	82.69	0.57	0.10	0.43
20x10S30	13.98	9.73	9.60	0.04	435737	time	11.38	0.03	98.46	0.91	0.00	0.11
Average	8.54	5.30	1.18	0.01			6.39	0.04		0.12	0.09	0.14

increasing the CPU time. The primal heuristic is able to find the optimal solution in 41 out of the 45 instances for which the optimal solution is known. When compared with the heuristics of Guignard et al. [28], it can obtain the optimal solution for 11 instances for which none of these heuristics is able to. Moreover, among all the 50 instances, the LR is able to provide the best known solution in 41 instances, as compared to 29 in LS1, 29 in LS2, and 28 in CHR. Finally, the LR average deviation is 0.04%, as compared to 0.12% for LS1, 0.09% for LS2, and 0.14% for CHR.

In the second part of the experiments, we compare the results obtained with formulations P2 and P3 when used with CPLEX and our LR algorithm with a new data set containing larger-size instances. We randomly generate a total of 60 instances which are divided into three sets containing 20 instances each. The three sets of instances differ in terms of the density of the flow matrix, which is considered to be 25%, 50%, and 75%, respectively. We use the same cardinality for the number of inbound and outbound doors as well as for the number of origins and destinations, respectively. The number of considered origins/destinations is $\{10, 15, 20, 25, 50\}$ and the number of considered inbound/outbound doors is $\{4, 6, 10, 20, 30\}$. The capacities and distances are generated in the same way as in Guignard et al. [28]. The considered values for the capacity slackness are 10%, %20 and %30. The flow matrix is generated using the procedure described in Boysen et al. [11]. In particular, the flow is randomly distributed according to a discrete uniform distribution and generated as $s_m \in [min, max]$ for each $m \in M$ and $r_n \in [min, max]$, for each $n \in N$ such that $\sum_{m\in M} s_m = \sum_{n\in N} r_n$. The parameters are defined as $min = 10|M|$ and $max = min \times density$.

The detailed results of the comparison using these three sets are given in Tables 2.3, 2.4 and 2.5, respectively. As in the previous table, the columns under the headings %gap, %dev, nodes and time present the results obtained based on formulation P2. Whenever CPLEX cannot solve the LP relaxation of a formulation or cannot find a feasible solution, we write NA in the corresponding entry. The symbol - denotes that an optimality gap cannot be computed with the information obtained by CPLEX within the CPU time limit.

The results from Tables 2.3, 2.4 and 2.5 show the limitations of using formulation P2 with CPLEX when solving larger instances. In particular, P2 is able to prove the optimality of the obtained solutions in 14 out of the 60 considered instances. In 14 instances CPLEX is not able to find a feasible solution within the CPU time limit and the average optimality gaps for the rest of the instances in each of the three sets are 9.81%, 10.70%, and 13.65%, respectively. As can be

Table 2.3: Results for Instances with 25% Density in Flow Matrix

	CPLEX						Lagrangean Relaxation		
Instance	% LP$_{P2}$	% LP$_{P3}$	% gap	% dev	nodes	time	% LR	% dev	time
10x4S10	4.61	3.09	0.00	0.00	1027	0.54	3.84	0.00	5.34
10x4S20	4.78	3.44	0.00	0.00	1752	1.24	4.44	0.00	6.38
10x4S30	3.76	3.27	0.00	0.00	867	0.24	3.76	0.00	6.36
15x6S10	6.75	4.92	0.00	0.00	388173	151.64	6.02	0.00	12.84
15x6S20	7.50	5.19	0.00	0.00	67858	35.30	6.52	0.00	18.46
15x6S30	6.16	4.51	0.00	0.00	38843	16.57	6.06	0.00	18.89
20x10S10	13.16	9.44	4.96	0.00	1870717	time	10.55	0.45	40.57
20x10S20	12.03	7.93	7.49	0.00	1777173	time	9.38	0.02	57.37
20x10S30	13.16	9.14	6.48	0.01	1731294	time	10.81	0.00	62.85
25x10S10	14.94	10.99	11.87	0.00	780843	time	12.07	0.23	78.39
25x10S20	13.34	9.45	11.92	0.00	671246	time	11.46	0.65	94.44
25x10S30	12.59	9.23	10.88	0.00	644769	time	10.92	0.02	115.33
50x10S10	18.77	14.97	18.02	2.54	49168	time	16.00	0.00	1013.89
50x10S20	16.63	13.79	16.01	2.54	55498	time	15.26	0.00	901.72
50x10S30	19.95	15.83	19.20	4.15	49380	time	16.54	0.00	705.01
50x20S10	32.17	NA	31.97	8.70	2671	time	26.52	0.00	1457.44
50x20S20	29.04	23.76	28.02	6.72	3590	time	24.87	0.00	1789.39
50x20S30	35.02	28.87	-	NA	0	time	27.88	0.00	858.13
50x30S20	44.29	NA	-	NA	0	time	36.12	0.00	1869.10
50x30S30	41.14	37.25	-	NA	0	time	34.37	0.00	2147.20
Average	17.49	11.95	9.81	1.45			14.67	0.07	

seen, the instances with larger densities in the flow matrix tend to have weaker LP bounds and thus, require more exploration in the branch-and-bound tree, increasing the required CPU time.

Table 2.4: Results for Instances with 50% Density in Flow Matrix

	CPLEX						Lagrangean Relaxation		
Instance	% LP$_{P2}$	% LP$_{P3}$	% gap	% dev	nodes	time	% LR	% dev	time
10x4S10	8.41	4.03	0.00	0.00	10060	6.30	5.05	0.00	9.74
10x4S20	5.98	3.66	0.00	0.00	761	0.63	4.86	0.00	13.05
10x4S30	6.60	3.85	0.00	0.00	2329	0.95	5.21	0.00	14.53
15x6S10	12.33	6.48	0.58	0.00	4287454	time	7.63	0.00	31.36
15x6S20	11.71	6.43	0.00	0.00	3456192	4432.88	8.30	0.00	34.23
15x6S30	10.39	6.05	0.00	0.00	731055	882.51	7.91	0.00	37.27
20x10S10	21.52	11.67	19.31	0.80	397942	time	13.43	0.00	90.02
20x10S20	19.03	10.26	16.74	1.80	401218	time	12.56	0.00	118.29
20x10S30	18.02	10.59	15.54	0.20	474253	time	13.35	0.00	134.51
25x10S10	21.70	12.10	20.15	1.77	203796	time	14.16	0.00	168.93
25x10S20	20.16	11.55	18.83	1.24	202228	time	13.88	0.00	204.44
25x10S30	17.93	10.88	16.20	0.53	219206	time	13.61	0.00	219.72
50x10S10	22.67	13.93	22.02	2.67	11296	time	15.94	0.00	1777.02
50x10S20	21.05	13.75	20.36	3.24	15020	time	15.82	0.00	2067.30
50x10S30	19.86	12.48	-	NA	0	time	15.05	0.00	2323.33
50x20S10	37.96	32.97	-	NA	0	time	26.48	0.00	2783.06
50x20S20	35.79	31.04	-	NA	0	time	25.56	0.00	3436.90
50x20S30	33.90	29.51	-	NA	0	time	25.40	0.00	4489.52
50x30S20	46.56	43.52	-	NA	0	time	33.47	0.00	5014.49
50x30S30	43.51	40.79	-	NA	0	time	32.86	0.00	6390.98
Average	21.75	15.78	10.70	0.87			15.53	0.00	

In terms of the LR, the primal heuristic is able to obtain a solution that is at least as good or better than the one obtained by CPLEX in 55 out of the 60 considered instances. Only in 5 instances from the first set with 25% density in the flow matrix CPLEX is able to obtain a better solution than LR. In 38 out of the 60 considered instances the LR is able to obtain a strictly better solution than CPLEX. The primal heuristic can obtain high quality solutions for the three sets of considered

Table 2.5: Results for Instances with 75% Density in Flow Matrix

	CPLEX						Lagrangean Relaxation		
Instance	% LP1	% LP2	% gap	UB % dev	nodes	time	% LR	% H dev	time
10x4S10	9.96	2.98	0.00	0.00	10517	7.98	4.51	0.00	11.51
10x4S20	9.20	4.06	0.00	0.00	8621	4.30	5.56	0.00	11.95
10x4S30	8.10	4.04	0.00	0.00	10495	5.38	6.03	0.00	14.12
15x6S10	14.87	4.89	5.07	0.06	4260595	time	7.12	0.00	30.44
15x6S20	13.64	4.77	7.95	0.00	3352373	time	7.39	0.00	26.64
15x6S30	12.09	5.28	3.04	0.00	3692989	time	7.83	0.00	46.69
20x10S10	23.81	9.23	20.47	0.44	267733	time	11.81	0.00	118.62
20x10S20	21.68	8.63	19.22	0.38	263216	time	12.85	0.00	146.29
20x10S30	20.89	9.12	19.49	1.17	341228	time	12.90	0.00	190.58
25x10S10	24.04	9.52	22.80	0.76	97156	time	12.79	0.00	225.95
25x10S20	22.36	8.54	21.06	1.39	97444	time	12.78	0.00	269.14
25x10S30	20.30	8.17	19.09	2.31	75710	time	12.63	0.00	292.09
50x10S10	24.44	10.15	23.75	2.88	3781	time	14.03	0.00	1733.01
50x10S20	22.68	9.89	21.87	2.50	3813	time	14.43	0.00	2424.74
50x10S30	21.61	NA	20.86	1.64	4054	time	14.24	0.00	3440.63
50x20S10	26.84	NA	-	NA	0	time	13.69	0.00	1181.36
50x20S20	37.54	34.13	-	NA	0	time	22.92	0.00	2615.49
50x20S30	35.94	NA	-	NA	0	time	23.16	0.00	3373.11
50x30S20	48.31	NA	-	NA	0	time	29.44	0.00	3925.52
50x30S30	46.43	NA	-	NA	0	time	28.72	0.00	6095.02
Average	23.24	8.89	13.65	0.90			13.74	0.00	

instances with an average deviation of 0.07%, 0.00% and 0.00%, respectively. Moreover, the LR algorithm always obtains a % LR gap smaller than the % gap of CPLEX in considerable less CPU time for the larger instances with 50 origins and destinations.

2.6 Conclusions

In this paper we have presented a linear MIP formulation for the cross-dock door assignment problem. Given the large number of variables and constraints in this formulation, we developed a Lagrangean relaxation algorithm that exploits the structure of the problem to obtain lower and upper bounds on the optimal solution value. Computational results confirm the efficiency and scalability of the proposed MIP formulation and solution approach. In particular, the MIP formulation can prove optimality of the obtained solutions of five benchmark instances for which the optimal solution was not known before. In addition, the LR algorithm provides high quality solutions for all the considered instances with up to 50 origins and destinations and 30 inbound and outbound doors in small CPU times with a performance guarantee. Given the inherent complexity of the non-linearities arising in the objective function of cross-dock door assignment problems, the optimal solution of larger-size instances of realistic size remains challenging. The authors are currently investigating the use of more sophisticated methodologies, such as column generation and bundle methods, for solving the Lagrangean dual problem and for improving the linear programming relaxation bounds.

Chapter 3

A Comparison of Formulations and Relaxations for the Cross-dock Door Assignment Problems

This paper deals with cross-dock door assignment problems in which the assignments of incoming trucks to strip doors, and outgoing trucks to stack doors are determined, with the objective of minimizing the total handling cost. We present two new mixed integer programming formulations which are theoretically and computationally compared with existing ones. One of such requires a column generation algorithm to solve its associated linear relaxation. We present the results of a series of computational experiments to evaluate the performance of the formulations on a set of benchmark instances. We also analyze the impact of the input-data on the quality of the associated linear programming relaxation bounds.

The following paper was submitted to *Computers and Operations Research* in March 2017. Nassief et al. [44].

3.1 Introduction

Cross-docking is a logistics strategy that facilitates rapid movement of consolidated products between suppliers and retailers within a supply chain. It is also a warehousing strategy that aims to reduce or ultimately eliminate storage and order picking, two of which are known to be major costly operations of any typical warehouse. This strategy has been used in the retailing, manufacturing, automotive, and photographic industries, and has been successfully implemented by several companies such as Walmart and Toyota. For examples of successful cross-docking implementations, the interested reader is referred to Forger [25], Kinnear [31], Witt [60], Chen and Song [16], and Napolitano [41].

Cross-dock facilities (or cross-docks) are designed to expedite the movement of highly and constantly demanded materials, which eventually lead to a better service level and quicker response across the supply chain at a reduced cost. Particularly, once incoming and outgoing trucks are assigned to their designated doors, the goods get unloaded from the incoming trucks, consolidated in a staging area according to their destinations, and loaded into the outgoing trucks with minimal storage in between. In practice, minimal storage is possible, and so, it comes in different forms that can range from temporary buffers in front of the doors to even a designated temporary storage area. Typically, the amount of time goods spend at a cross-dock ranges from virtually none to about 24 hours. The various advantages of cross-docking as compared to the different practices of traditional warehouses and point-to-point deliveries are listed in Van Belle et al. [58].

Given the inherent complexity of designing and operating cross-docks, several classes of decision problems have been studied in the literature. Agustina et al. [2], Boysen [10], Shuib and Fatthi [53], Van Belle et al. [58], and Buijs et al. [14] provide reviews of decision problems arising in cross-docking. Moreover, a recent review by Ladier and Alpan [32] proposes a framework that highlights the gaps between the literature and some cross-docking practices in France.

In this paper we focus on a class of operational problems referred to as *cross-dock door assignment problems* (CDAPs). On a daily basis, fully loaded incoming trucks arrive to the cross-dock facility. When docked (or assigned) to strip (inbound) doors, they get unloaded by employees who inspect and sort the shipments according to their destinations. Using material handling equipment such as carts, forklifts or a system of conveyors, these shipments are transferred to some stack (outbound) doors ready for loading into outgoing trucks, which are assigned according to their destinations. The material handling equipment keep traveling between strip and stack doors till all products are transferred to their designated stack doors. Then, employees load them into outgoing trucks. Whenever an outgoing truck is filled with all needed products for its designated destination, it leaves the facility carrying the consolidated products. CDAPs seek to optimally decide on the assignment of both incoming trucks to strip doors and outgoing trucks to stack doors such that the total handling cost inside the cross-dock is minimized. The assignment decisions affect the total time of getting shipments unloaded, traveled across the facility, and loaded. The cost is commonly measured as traveling distance between doors and thus, the objective needs only to focus on the weighted travel time between doors. Tsui and Chang [55, 56], Oh et al. [45], Bozer and Carlo [13], Cohen and Keren [21], Zhu et al. [62], Guignard et al. [28], and Nassief et al. [42] study CDAPs with this type of objective. However, in practice unloading and loading times do need to be taken into account, specially when these depend on the number and skill-level of workers assigned to each door. This is particularly relevant in applications where turnovers are very high Amini et al. [see, for instance 5] and thus, there is learning curve to be taken into account when unloading and loading. To the best of our knowledge, there are only two works explicitly considering unloading and loading times at doors. Peck [46] incorporates unloading and loading times along with travel times as capacity constraints. Zhang et al. [61] introduce these terms in one objective function in the context of truck scheduling. The authors report on the impact of considering one term at a time versus all terms in a single objective function. We refer to Nassief et al. [42] for a detailed literature review on CDAPs and Gelareh et al. [26] and Enderer et al. [23] for recent extensions of CDAPs in which additional operational decisions are considered.

In this paper we study the standard CDAP considered in Zhu et al. [62], Guignard et al. [28], and Nassief et al. [42] and show how door-dependent unloading and loading costs can be easily integrated in the objective, in addition to the transfer costs between doors. We assume workforce assignment decisions to be exogenous, i.e., these are not part of the decision process. From now on, we refer to this problem as the *cross-dock door assignment problem* (CDAP). Given the inherent complexity of the quadratic nature of CDAPs, most of the previous studies have resorted to heuristic algorithms for their solution. However, little work has been done for the study and development of mathematical programming formulations that can be solved with general purpose solvers or embedded into decomposition techniques. The main contribution of this work is to study several mixed integer programming (MIP) formulations for the CDAP and to report on their performance. In particular, we present two new MIP formulations which are theoretically and computationally compared with existing formulations of the CDAP. The formulations are compared with respect to the quality of their linear programming (LP) relaxation bounds and with respect to the Lagrangean relaxation introduced in Nassief et al. [42]. Due to the huge number of variables involved in one the new formulations, we implement a column generation algorithm to solve its LP relaxation. Finally, we present the results of a series of computational experiments to evaluate the relative performance of the proposed and existing formulations and of the decomposition algorithms. As will be shown, although some formulations provide the same theoretical bound, the convergence of their solution scheme may vary significantly.

The remainder of this paper is structured as follows. Section 3.2 provides a formal definition of the CDAP. In Section 3.3, we introduce several linear MIP formulations. Section 3.4 presents the theoretical comparisons of the bounds provided by their LP relaxations and the Lagrangean relaxation of Nassief et al. [42]. In Section 3.5, we introduce the column generation algorithm used to solve one of the MIP formulations. Computational experiments using a set of benchmark instances are presented in Section 3.6. Conclusions follow in Section 3.7.

3.2 Problem Statement

Let M, N, I, and J denote the set of origins, destinations, inbound and outbound doors, respectively. The traveling time (or distance) between strip door i and stack door j is denoted by t_{ij}. Let u_i and ℓ_j denote the per unit unloading and loading times for inbound door i and outbound door j, respectively. Let w_{mn} be the amount of commodity originated at m and destined to n. Without loss of generality, the values w_{mn} can be stated in terms of the number of times the material handling equipment (i.e., forklift, pallet jack, etc) needs to be used to transfer all the flow originated at m with destination n between a pair of strip/stack doors. For each $m \in M$, let $s_m = \sum_{n \in N} w_{mn} > 0$ denote the total supplied flow from an origin whereas for each $n \in N$, let $r_n = \sum_{m \in M} w_{mn} > 0$ be the total sent flow towards a destination. We denote by S_i and R_j the capacity for inbound door $i \in I$ and outbound door $j \in J$, respectively. This capacity represents the limit on the amount of commodities that the workforce and material handling equipment designated to each door can process in a given shift. The CDAP consists of assigning each origin and each destination to exactly one inbound door and one outbound door, respectively, such that the capacity restrictions on dock doors are satisfied and the total handling cost (unloading, transfer and loading) inside the cross-dock is minimum.

Given that each commodity has to pass through exactly one inbound and one outbound door, origin/destination paths are of the form (m, i, j, n), where m is the origin, i, j the door pair, and n the destination. The handling cost of path (m, i, j, n) for routing w_{mn} is then computed as $w_{mn}\,(u_i + t_{ij} + \ell_j)$. We would like to highlight that previous papers dealing with the CDAP do not consider explicitly the unloading and loading times and only focus on the transfer time between doors. That is, the handling cost is computed as $w_{mn}t_{ij}$.

A *natural* way to formulate the CDAP is to consider it as two *generalized assignment problems* linked by a quadratic cost associated with the interaction of inbound and outbound doors. For each pair $m \in M, i \in I$, we define inbound assignment variables x_{mi} equal to 1 if and only if origin m is assigned to inbound door i. Similarly, for each pair $n \in N$, $j \in J$, we define outbound assignment variables y_{nj} equal to 1 if and only if destination n is assigned to outbound door j. Using these sets of variables, the CDAP can be formulated as the following bilinear integer program Zhu et al. [62]:

$$
\begin{aligned}
[P0] \quad \text{minimize} \quad & \sum_{m\in M}\sum_{n\in N}\sum_{i\in I}\sum_{j\in J} w_{mn}\,(u_i + t_{ij} + \ell_j)\,x_{mi}y_{nj} && && (3.1)\\
\text{subject to} \quad & \sum_{i\in I} x_{mi} = 1 && m \in M && (3.2)\\
& \sum_{j\in J} y_{nj} = 1 && n \in N && (3.3)\\
& \sum_{m\in M} s_m x_{mi} \le S_i && i \in I && (3.4)\\
& \sum_{n\in N} r_n y_{nj} \le R_j && j \in J && (3.5)\\
& x_{mi} \in \{0,1\} && m \in M, i \in I && (3.6)\\
& y_{nj} \in \{0,1\} && n \in N, j \in J. && (3.7)
\end{aligned}
$$

The objective function minimizes the total unloading time of incoming shipment, weighted time traveled by the material handling equipment, and loading time of outgoing shipment. Constraints (3.2) and (3.3) guarantee that every origin (destination) is assigned to exactly one inbound (outbound) door. Constraints (3.4) and (3.5) ensure that the capacity constraints on the inbound and outbound doors, respectively, are satisfied. Constraints (3.6) and (3.7) are the classical integrality conditions on the decision variables. Observe that constraints (3.2)-(3.7) define the set of feasible solutions of two independent generalized assignment problems and the quadratic term in (3.1) links them.

The objective function (3.1) can be alternatively written as

$$\text{minimize} \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{m\in M}\sum_{n\in N}\sum_{i\in I}\sum_{j\in J} w_{mn} t_{ij} x_{mi} y_{nj} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj}. \tag{3.8}$$

The first and last term compute the unloading and loading costs, respectively, independently of the transfer cost. Even though both objective functions always provide the same evaluation at integer feasible solutions, this may not be the case for fractional solutions. Specially, when linearizing or substituting the nonlinear term for the objective (3.1) using additional decision variables. It is known that, for the closely related quadratic assignment problem, lower bounds can be tightened by moving as much information as possible form the quadratic term to the linear term Li et al. [34]. Therefore, from now on we will use objective (3.8) instead of (3.1).

3.3 Mixed-integer Programming Formulations

The objective of CDAP is to minimize the total handling cost of unloading, transferring between pairs of doors, and loading commodities inside the cross-dock. From $P0$, we note that knowing the assignment pattern of origins and destinations to inbound and outbound doors, respectively, is sufficient to evaluate the quadratic term in objective function. However, in order to state the objective as a linear function additional decision variables are needed to model the paths that commodities follow. We next present new and existing linear MIP formulations based on different approaches to model such O/D paths. The existing formulations of the CDAP rely on the use of binary variables to model the set of paths between origins and destinations. One of the new formulations use continuous variables to compute the amount of flow transversing each door pair. Finally, the third one uses a compact definition of variables to model assignment configurations between origins and inbound doors and destinations and outbound doors.

3.3.1 Path-based Formulations

One way to model the O/D paths at cross-docks is to define path-based variables commonly used in multi-commodity network design problems. To simplify the notation, we denote by K the set of commodities whose origin and destination points belong to M and N, respectively. For each commodity $k \in K$, w_k is the amount of commodity k to be routed from origin $o(k) \in M$ to destination $d(k) \in N$.

The following formulation was originally introduced in Nassief et al. [42] for the CDAP. For each $k \in K$, $i \in I$, $j \in J$, we define binary variables z_{kij} equal to 1 if and only if commodity k transits via inbound door i and outbound door j. Using the z_{kij} variables and the x_{mi} and y_{nj} variables previously defined, the CDAP can be formulated as:

$$[P1] \quad \text{minimize} \quad \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj}$$

subject to (3.4) – (3.7),

$$\sum_{i\in I}\sum_{j\in J} z_{kij} = 1 \qquad k \in K \tag{3.9}$$

$$\sum_{j\in J} z_{kij} = x_{o(k)i} \qquad k \in K, i \in I \tag{3.10}$$

$$\sum_{i\in I} z_{kij} = y_{d(k)j} \qquad k \in K, j \in J \tag{3.11}$$

$$z_{kij} \geq 0 \qquad k \in K, i \in I, j \in J. \tag{3.12}$$

Constraints (3.9) ensure that each commodity k must go through one unique pair of doors $i - j$. Constraints (3.10) state that if the origin of commodity k is assigned to inbound door i, then it must be routed through inbound door i and some outbound door j. Similarly, constraints (3.11) state that if the destination of commodity k is assigned to an outbound door j then it must be routed through outbound door j and some inbound door i. Finally, constraints (3.12) are the standard nonnegativity conditions on the decision variables. Note that there is no need to explicitly state the integrality conditions on the z_{kij} variables given that equalities (3.10)–(3.11) have no feasible fractional solution whenever x_{mi} and y_{nj} variables take a binary value.

A slightly different path-based formulation can be obtained by replacing constraints (3.9) in P1 with constraints (3.2)–(3.3) of P0. It can be written as follows:

$$[P2] \qquad \text{minimize} \quad \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj}$$

$$\text{subject to} \quad (3.2)-(3.7),(3.10)-(3.12).$$

Both P1 and P2 can be strengthened by adding the following sets of valid inequalities:

$$\sum_{k\in K:d(k)=n} s_{o(k)} z_{kij} \leq S_i y_{nj} \qquad n\in N, j\in J, i\in I \tag{3.13}$$

$$\sum_{k\in K:o(k)=m} r_{d(k)} z_{kij} \leq R_j x_{mi} \qquad m\in M, i\in I, j\in J. \tag{3.14}$$

Constraints (3.13) state that if a commodity originated at node m with destination n is being routed through inbound door i, then all commodities originated at the same origin have to be routed trough the same inbound door regardless of their destination. Similarly, (3.14) state that if a commodity destined at node n with origin m is being routed through outbound door j, then all commodities destined at the same destination have to be routed trough the same outbound door regardless of their origin. From now on, we refer to the extended path-based formulations of P1 and P2 that include constraints (3.13) and (3.14) as P1′ and P2′, respectively. Nassief et al. [42] show that these inequalities, although redundant for P1, play an important role in improving its linear programming (LP) relaxation bound. Finally, we note that these inequalities can also be derived by using the *reformulation linearization technique* of Sherali and Adams [52].

3.3.2 A Flow-based Formulation

The CDAP can be seen as a particular case of a multi-commodity network design problem. Thus, it can be modeled using the so-called flow-based variables to compute the amount of flow originated at m routed between $i-j$ doors. For each $m\in M$, $i\in I$, $j\in J$ we define continuous variables z'_{mij} equal to the amount of flow originated at m routed from inbound door i to outbound door j. Using these variables, we introduce a flow-based formulation of the CDAP problem as follows:

$$[F1] \qquad \text{minimize} \quad \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{m\in M}\sum_{i\in I}\sum_{j\in J} t_{ij} z'_{mij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj}$$

$$\text{subject to} \quad (3.2) - (3.7) \qquad (3.15)$$

$$\sum_{j\in J} z'_{mij} = s_m x_{mi} \qquad m\in M, i\in I \qquad (3.16)$$

$$\sum_{i\in I} z'_{mij} = \sum_{n\in N} w_{mn} y_{nj} \qquad m\in M, j\in J \qquad (3.17)$$

$$z'_{mij} \geq 0 \qquad m\in M, i\in I, j\in J. \qquad (3.18)$$

Constraints (3.16) and (3.17) are the flow conservation constraints for inbound and outbound doors. In particular, constraints (3.16) ensure that if an origin m is assigned to inbound door i, then the total flow coming from origin m must be exactly the same as the total flow split through all outbound doors j. Similarly, (3.17) guarantee that for a given origin m and outbound door j, the total outgoing flow to destinations that are assigned to outbound door j is equal to the total incoming flow at outbound door j. Finally, constraints (3.18) are the standard nonnegativity conditions. It is easy to see that a symmetrically identical flow based formulation can be derived by redefining the flow-based variables to compute the amount of flow destined at n and routed between $i - j$ doors, and thus is omitted.

3.3.3 A Configuration-based Formulation

We next present a formulation for the CDAP which uses configuration-based variables to characterize the feasible subsets of origins and destinations that can be assigned to inbound and outbound doors, respectively.

Let C_i denote the set of origins' subsets, which are assigned to strip door i and satisfy the capacity constraints, i.e., $C_i = \{M' \subseteq M : \sum_{m\in M'} s_m \leq S_i\}$. Similarly, let H_j denote the set of destinations' subsets, which are assigned to stack door j and satisfy the capacity constraints, i.e., $H_j = \{N' \subseteq N : \sum_{n\in N'} r_n \leq R_j\}$. For each $i \in I$ and $c \in C_i$, we define binary variables x_i^c equal to 1 if and only if configuration c is selected for door i. For each $j \in J$ and $h \in H_j$, we define binary variables y_j^h equal to 1 if and only if configuration h is selected for door j. Using these two sets of configuration-based variables together with the z_{kij} variables defined in Section

3.3.1, the configuration-based formulation can be written as follows:

$$[DC] \quad \text{minimize} \quad \sum_{c \in C_i : o(k) \in c} \sum_{m \in M} \sum_{i \in I} s_m u_i x_i^c + \sum_{i \in I} \sum_{j \in J} w_k t_{ij} z_{kij} + \sum_{h \in H_j : d(k) \in h} \sum_{n \in N} \sum_{j \in J} r_n \ell_j y_j^h$$

$$\text{subject to} \quad (3.9), (3.12)$$

$$\sum_{j \in J} z_{kij} = \sum_{c \in C_i : o(k) \in c} x_i^c \qquad k \in K, i \in I \tag{3.19}$$

$$\sum_{i \in I} z_{kij} = \sum_{h \in H_j : d(k) \in h} y_j^h \qquad k \in K, j \in J \tag{3.20}$$

$$\sum_{c \in C_i} x_i^c = 1 \qquad i \in I \tag{3.21}$$

$$\sum_{h \in H_j} y_j^h = 1 \qquad j \in J \tag{3.22}$$

$$x_i^c \in \{0, 1\} \qquad i \in I, c \in C_i \tag{3.23}$$

$$y_j^h \in \{0, 1\} \qquad j \in J, h \in H_j. \tag{3.24}$$

Constraints (3.19) and (3.20) link the configuration-based variables with the path-based variables to ensure that commodities are properly routed. Constraints (3.21) and (3.22) state that exactly one configuration is selected for each inbound and outbound door, respectively. Finally, constraints (3.23) and (3.24) are the integrality restrictions on the configuration variables.

DC can be strengthened by adapting constraints (3.13) and (3.14), used in $P1'$ and $P2'$, as follows:

$$\sum_{k \in K : d(k) = n} s_{o(k)} z_{kij} \le S_i \sum_{h \in H_j : n \in h} y_j^h \qquad n \in N, j \in J, i \in I \tag{3.25}$$

$$\sum_{k \in K : o(k) = m} r_{d(k)} z_{kij} \le R_j \sum_{c \in C_i : m \in c} x_i^c \qquad m \in M, i \in I, j \in J. \tag{3.26}$$

As in $P1'$ and $P2'$, these valid inequalities, although redundant for DC, play an important role in improving its LP relaxation bound. Hence, we refer to DC' as the configuration-based formulation with the addition of these inequalities.

3.4 A Comparison of Relaxations

In this section, we analytically compare the quality of the lower bounds yielded by the LP relaxations of the MIP formulations presented in Section 3.3. We provide dominance or equivalence relationships between them. The LP relaxations of $P1$, $P2$, and $F1$, denoted as $LP1$, $LP2$, $LF1$, are obtained when removing the integrality conditions on the x_{mi} and y_{nj} variables. In the case of DC the LP relaxation is obtained when eliminating the integrality conditions on the x_i^c nd y_j^c variables. Respectively, we denote by z_P, z_F and z_{DC} the optimal values of the LP relaxations of formulations P, F and DC.

Proposition 1. $z_{P1} = z_{P2}$.

Proof. Given that the objective in $P1$ and $P2$ are the same, we only need to show that every feasible solution $(\hat{x}, \hat{y}, \hat{z})$ of $LP1$ is also a feasible solution of $LP2$, and vice versa. We first assume that $(\hat{x}, \hat{y}, \hat{z})$ is a feasible solution of $LP1$. For each $k \in K$, summing up (3.10) over all $i \in I$ and (3.11) over all $j \in J$, we obtain

$$\sum_{i \in I} \sum_{j \in J} \hat{z}_{kij} = \sum_{i \in I} \hat{x}_{o(k)i} \tag{3.27}$$

$$\sum_{i \in I} \sum_{j \in J} \hat{z}_{kij} = \sum_{j \in J} \hat{y}_{d(k)j}. \tag{3.28}$$

Now, substituting constraint $\sum\limits_{i \in I} \sum\limits_{j \in J} \hat{z}_{kij} = 1$ in (3.27) and (3.28), leads to

$$1 = \sum_{i \in I} \hat{x}_{o(k)i} \qquad k \in K$$

$$1 = \sum_{j \in J} \hat{y}_{d(k)j}. \qquad k \in K.$$

Then we have

$$\sum_{i \in I} \hat{x}_{mi} = 1 \qquad m \in M$$

$$\sum_{j \in J} \hat{y}_{nj} = 1 \qquad n \in N.$$

Similarly, assume now that $(\hat{x}, \hat{y}, \hat{z})$ is a feasible solution of $LP2$. For each $k \in K$, substituting $\sum\limits_{i \in I} \hat{x}_{o(k)i} = 1$ and $\sum\limits_{j \in J} \hat{y}_{d(k)j} = 1$ in (3.27) and (3.28), leads to

$$\sum_{i \in I} \sum_{j \in J} \hat{z}_{kij} = 1,$$

and the result follows. □

A very similar proof can be done to show that $z_{P1'} = z_{P2'}$ and it is thus omitted.

Proposition 2. $z_{F1} \leq z_{P2}$.

Proof. We will show that every feasible solution $(\hat{x}, \hat{y}, \hat{z})$ of $LP2$ is also a feasible solution of $LF1$. Summing up (3.10) over $k \in K$ such that $o(k) = m$ and multiplying by w_k, we obtain

$$\sum_{k \in K: o(k)=m} \sum_{j \in J} w_k \hat{z}_{kij} = \sum_{k \in K: o(k)=m} w_k \hat{x}_{mi} = s_m x_{mi} \tag{3.29}$$

for each $m \in M$, $i \in I$, where the second equality follows from $\sum_{k \in K: o(k)=m} w_k = s_m$ for each $m \in M$. Let

$$z'_{mij} = \sum_{k \in K: o(k)=m} w_k \hat{z}_{kij}. \tag{3.30}$$

Summing over j in (3.30) and using (3.29) leads to:

$$\sum_{j \in J} z'_{mij} = \sum_{j \in J} \sum_{k \in K: o(k)=m} w_k \hat{z}_{kij} = s_m \hat{x}_{mi} \qquad m \in M, i \in I,$$

which corresponds to constraints (3.16).

We now sum up (3.11) over $k \in K$ such that $o(k) = m$. Multiplying by w_k, we obtain

$$\sum_{k \in K: o(k)=m} \sum_{i \in I} w_k \hat{z}_{kij} = \sum_{k \in K: o(k)=m} w_k \hat{y}_{d(k)j} = \sum_{n \in N} w_{mn} \hat{y}_{nj}, \tag{3.31}$$

for each $m \in M$, $j \in J$. Summing over j in (3.30) and using (3.31) leads to:

$$\sum_{i \in I} z'_{mij} = \sum_{n \in N} w_{mn} \hat{y}_{nj} \qquad m \in M, j \in J,$$

which corresponds to constraints (3.17) and the result follows. □

Proposition 3. $z_{P1} \leq z_{DC}$.

Proof. In $P1$, if we relax constraints (3.10) and (3.11) using Lagrangean relaxation, weighting their violations with a multiplier vector (μ, ν) of appropriate dimension, we obtain the following Lagrangean function:

$$\begin{aligned} L(\mu,\nu) = \min_{(z,x,y)} \Bigg\{ & \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj} \\ & + \sum_{k\in K}\sum_{i\in I} \mu_{ki} \left(\sum_{j\in J} z_{kij} - x_{o(k)i} \right) \\ & + \sum_{k\in K}\sum_{j\in J} \nu_{kj} \left(\sum_{i\in I} z_{kij} - y_{d(k)j} \right) : \text{(3.4)} - \text{(3.9)}, \text{(3.12)} \Bigg\}, \end{aligned}$$

and its associated Lagrangean dual problem

$$z_{LD} = \max_{\mu,\nu} L(\mu,\nu).$$

It is then equivalent to the following linear program Geoffrion [27]:

$$\begin{aligned} [PR] \quad \text{minimize} \quad & \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj} \\ \text{subject to:} \quad & \sum_{j\in J} z_{kij} = x_{o(k)i} \qquad k\in K, i\in I \qquad \text{(3.10)} \\ & \sum_{i\in I} z_{kij} = y_{d(k)j} \qquad k\in K, j\in J \qquad \text{(3.11)} \\ & (x,y,z) \in Co \Bigg\{ (x,y,z) \in B^{|M||I|} \times B^{|N||J|} \times R_+^{|K||I||J|} : \\ & \qquad \sum_{i\in I}\sum_{j\in J} z_{kij} = 1,\ k\in K, \\ & \qquad \sum_{m\in M} s_m x_{mi} \le S_i,\ i\in I, \sum_{n\in N} r_n y_{nj} \le R_j,\ j\in J \Bigg\}, \quad \text{(3.32)} \end{aligned}$$

where $Co\{X\}$ denotes the convex hull of X.

Taking into account that the convex hull in (3.32) is the intersection of several independent polytopes $Z \cup X \cup Y$, where $Z = \bigcup_{k\in K} Z_k$, $X = \bigcup_{i\in I} X_i$, $Y = \bigcup_{j\in J} Y_j$, and

$$\begin{aligned} Z_k = \quad & Co\left\{ z_k \in R_+^{|I||J|} : \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 \right\}, \\ X_i = \quad & Co\left\{ x_i \in B^{|M|} : \sum_{m\in M} s_m x_{mi} \le S_i \right\}, \\ Y_j = \quad & Co\left\{ y_j \in B^{|N|} : \sum_{n\in N} r_n y_{nj} \le R_j \right\}, \end{aligned}$$

Note that $Z_k = \left\{ z_k \in R_+^{|I||J|} : \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 \right\}$ because of total unimodularity. PR is then equivalent to the linear program:

$$[PR'] \quad \text{minimize} \quad \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj}$$

$$\text{subject to:} \quad \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 \qquad k \in K \tag{3.9}$$

$$\sum_{j\in J} z_{kij} = x_{o(k)i} \qquad k \in K, i \in I \tag{3.10}$$

$$\sum_{i\in I} z_{kij} = y_{d(k)j} \qquad k \in K, j \in J \tag{3.11}$$

$$x_i \in Co\left\{ x_i \in B^{|M|} : \sum_{m\in M} s_m x_{mi} \le S_i \right\} \qquad i \in I$$

$$y_j \in Co\left\{ y_j \in B^{|N|} : \sum_{n\in N} r_n y_{nj} \le R_j \right\} \qquad j \in J$$

$$z_{kij} \ge 0 \qquad k \in K, i \in I, j \in J. \tag{3.12}$$

For each $i \in I$, $j \in J$, let $E(X_i)$ and $E(Y_j)$ denote the set of extreme points of X_i and Y_j, respectively. Given that any point in a polytope can be written as a convex combination of its extreme points, PR' can be restated as the following linear program:

$$[PR''] \quad \text{minimize} \quad \sum_{c\in E(X_i):o(k)\in c}\sum_{m\in M}\sum_{i\in I} s_m u_i x_i^c + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{h\in E(Y_j):d(k)\in h}\sum_{n\in N}\sum_{j\in J} r_n \ell_j y_j^h$$

$$
\begin{aligned}
\text{subject to}\quad & \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 && k\in K\\
& \sum_{j\in J} z_{kij} = \sum_{c\in E(X_i):o(k)\in c} x_i^c && k\in K, i\in I\\
& \sum_{i\in I} z_{kij} = \sum_{h\in E(Y_j):d(k)\in h} y_j^h && k\in K, j\in J\\
& \sum_{c\in E(X_i)} x_i^c = 1 && i\in I\\
& \sum_{h\in E(Y_j)} y_j^h = 1 && j\in J\\
& x_i^c \geq 0 && i\in I, c\in E(X_i)\\
& y_j^h \geq 0 && j\in J, h\in E(Y_j)\\
& z_{kij} \geq 0 && k\in K, i\in I, j\in J.
\end{aligned}
$$

Finally, given that the set of extreme points $E(X_i)$ and $E(Y_j)$ are contained in the set of feasible configurations $C_i = \{M' \subseteq M : \sum_{m\in M'} s_m \leq S_i\}$ and $L_j = \{N' \subseteq N : \sum_{n\in N'} r_n \leq R_j\}$, respectively, and that every configuration $c \in C_i$ and $h \in H_j$ can be written as a convex combination of points in $E(X_i)$ and $E(Y_j)$, respectively, then PR'' is equivalent to the LP relaxation of DC and thus, we have $z_{P1} \leq z_{LD} = z_{DC}$, and the result follows. As will be shown in Section 3.6, z_{P1} is often smaller than z_{DC}. This is explained by the fact that the Lagrangean dual problem does not have the integrality property, it follows that $z_{P1} \leq z_{LD}$. $\square$

We now present a simple combinatorial bound for the CDAP which can be obtained by using the information on the minimum handling cost of unloading, loading and transferring commodities between door pairs for every $k \in K$. In particular, when relaxing the linking constraints (3.10), (3.11) and the capacity constraints (3.5), (3.4) from P2 and adding constraints (3.9), we obtain the following combinatorial relaxation:

$$
\begin{aligned}
(COMB) \quad z_{COMB} = \text{minimize} \quad & \sum_{m\in M}\sum_{i\in I} s_m u_i x_{mi} + \sum_{k\in K}\sum_{i\in I}\sum_{j\in J} w_k t_{ij} z_{kij} + \sum_{n\in N}\sum_{j\in J} r_n \ell_j y_{nj} \\
\text{subject to} \quad & \sum_{i\in I}\sum_{j\in J} z_{kij} = 1 \qquad k \in K \\
& \sum_{i\in I} x_{mi} = 1 \qquad m \in M \\
& \sum_{j\in J} y_{nj} = 1 \qquad n \in N \\
& x_{mi} \in \{0,1\} \qquad m \in M, i \in I \\
& y_{nj} \in \{0,1\} \qquad n \in N, j \in J \\
& z_{kij} \geq 0 \qquad k \in K, i \in I, j \in J.
\end{aligned}
$$

Proposition 4. $z_{COMB} \leq z_{F1}$.

Proof. $COMB$ can be decomposed into three independent subproblems, one for each set of x, y, and z variables, respectively. The subproblems in the space of the x and y variables can be further decomposed into several independent semi-assignment problems which can be solved by inspection. Given that the cost t_{ij} does not depend on commodity k, the optimal solution of the third subproblem is given by the path having the shortest cost. Therefore, we have

$$z_{COMB} = \sum_{m\in M} s_m u_i^{min} + \sum_{k\in K} w_k t_{ij}^{min} + \sum_{n\in N} r_n \ell_j^{min},$$

where $u_i^{min} = \min\{u_i : i \in I\}$, $t_{ij}^{min} = \min\{t_{ij} : i \in I, j \in J\}$, and $\ell_j^{min} = \min\{\ell_j : j \in J\}$. Consider now a relaxation of $F1$, denoted as R$F1$, where all constraints but (3.2), (3.3), and (3.16) are relaxed. Summing up (3.16) over $i \in I$, we obtain

$$\sum_{i\in I}\sum_{j\in J} z'_{mij} = s_m \sum_{i\in I} x_{mi} = s_m,$$

where the last equality follows from (3.2). Thus, an optimal solution to $RF1$ is given by routing the flow originated at each origin $m \in M$ through the door pair having the smallest unloading and transfer cost, i.e.,

$$z_{RF1} = \sum_{m\in M} s_m \min_{i\in I, j\in J}\{u_i + t_{ij}\} + \sum_{n\in N} r_n \ell_j^{min}.$$

Therefore, we have $z_{COMB} \leq z_{RF1} \leq z_{F1}$, and the result follows. □

The following corollary summarizes the results from this section.

Corollary 1. $z_{COMB} \leq z_{F1} \leq z_{P1} = z_{P2} \leq z_{DC} \leq z_{DC'}$.

This means that in terms of the LP relaxation bounds, the best theoretical bound is based (DC'), whereas the flow-based formulation $(F1)$, has the weakest lower bound. In Section 3.6, we present the results of computational experiments to compare the quality of the LP bounds of all formulations along with the combinatorial and Lagrangean bounds.

3.5 A Column Generation Algorithm

Column generation (CG) is a decomposition method used to solve linear programs with an enormous number of variables. The main idea of the method is to divide the original linear program, denoted as the *master problem* (MP), into two interrelated subproblems: a *restricted master problem* (RMP) and a set of *pricing problems* (PPs). The RMP contains only a small subset of the columns (or variables) of the original MP. At every iteration, the RMP is solved to optimality and additional columns are added on the fly. The PPs are solved to determine whether the current solution of the RMP is optimal for the MP or to identify additional columns with negative reduced costs to be added to the RMP. This process is repeated as long as new columns with negative reduced costs can be identified or terminates when an ε-optimal solution for the MP has been identified.

We next present a CG algorithm to solve the LP relaxation of the configuration-based formulation DC'. In addition, initial columns and termination criterion are explained briefly.

3.5.1 The Restricted Master Problem

Let C_i^t denote the subset of configurations for inbound door i at iteration t, H_j^t denote the subset of feasible configurations for outbound door j at iteration t, and D_k^t the subset of available door pairs for commodity k at iteration t. The RPM at iteration t can be stated as follows:

$$[RMP^t] \quad \text{minimize} \quad \sum_{c \in C_i^t} \sum_{m \in M} \sum_{i \in I} s_m u_i a_c x_c + \sum_{k \in K} \sum_{(i,j) \in D_k^t} w_k d_{ij} z_{kij} + \sum_{h \in H_j^t} \sum_{n \in N} \sum_{j \in J} r_n \ell_i b_h y_h$$

$$\text{subject to:} \quad \sum_{(i,j)\in D_k^t} z_{kij} = 1 \qquad k \in K \tag{3.33}$$

$$\sum_{(i,j)\in D_k^t} z_{kij} = \sum_{c\in C_i^t} a_c x_c \qquad k \in K, i \in I \tag{3.34}$$

$$\sum_{(i,j)\in D_k^t} z_{kij} = \sum_{h\in H_j^t} b_h y_h \qquad k \in K, j \in J \tag{3.35}$$

$$\sum_{k\in K} \sum_{(i,j)\in D_k^t} s_{o(k)} z_{kij} \le S_i \sum_{h\in H_j^t : n\in h} y_h \qquad n \in N, j \in J, i \in I \tag{3.36}$$

$$\sum_{k\in K} \sum_{(i,j)\in D_k^t} r_{d(k)} z_{kij} \le R_j \sum_{c\in C_i^t : m\in c} x_c \qquad m \in M, i \in I, j \in J \tag{3.37}$$

$$\sum_{c\in C_i^t} x_c = 1 \qquad i \in I \tag{3.38}$$

$$\sum_{h\in H_j^t} y_h = 1 \qquad j \in J \tag{3.39}$$

$$x_c \ge 0 \qquad c \in C_i^t \tag{3.40}$$

$$y_h \ge 0 \qquad h \in H_j^t \tag{3.41}$$

$$z_{kij} \ge 0 \qquad k \in K, (i,j) \in D_k^t. \tag{3.42}$$

Note that, although there is a polynomial number of routing variables z_{kij}, we do not add all of them as very few of them will be used in an optimal solution of the LP relaxation of DC'.

3.5.2 The Pricing Problem

Let $\mu_k^{(3.33)}$, $\mu_{ki}^{(3.34)}$, $\mu_{kj}^{(3.35)}$, $\mu_{nji}^{(3.36)}$, $\mu_{mij}^{(3.37)}$, $\mu_i^{(3.38)}$, and $\mu_j^{(3.39)}$ be the dual variables associated with constraints (3.33), (3.34), (3.35), (3.36), (3.37), (3.38), and (3.39), respectively. Given that the MP contains three sets of decision variables x, y, and z, there are three pricing problems, one for each type of the variables. In each CG iteration, we solve optimally all the three pricing problems, and add all columns with negative reduced costs.

The first pricing problem is associated with the configuration of inbound doors. It can be further decomposed into $|I|$ independent subproblems, one for each door:

$$[PP_i^X] \quad \text{minimize} \quad \sum_{m \in M} \left(s_m u_i + \mu_{ki:o(k)=m}^{(3.34)} + \sum_{j \in J} R_j \mu_{mij}^{(3.37)} \right) x_{mi} - \mu_i^{(3.38)}$$

$$\text{subject to:} \quad \sum_{m \in M} s_m x_{mi} \leq S_i \tag{3.43}$$

$$x_{mi} \in \{0,1\} \qquad m \in M, \tag{3.44}$$

where x_{mi} are the binary decision variables equal to 1 if and only if origin m is assigned to inbound door i. Note that the objective function implicitly evaluates the reduced cost of all feasible configurations in C_i. The optimal solution to PP_i^X is thus an inbound door configuration for i having the smallest reduced cost coefficient. PP_i^X is a 0-1 knapsack problem which is known to be NP-hard. However, it can be efficiently solved using the 0-1 knapsack algorithm given in Martello et al. [39].

The second pricing problem is associated with the configuration of outbound doors. It can also be further decomposed into $|J|$ independent subproblems, one for each door:

$$[PP_j^Y] \quad \text{minimize} \quad \sum_{n \in N} \left(r_n \ell_i + \mu_{kj:d(k)=n}^{(3.35)} + \sum_{i \in I} S_i \mu_{nji}^{(3.36)} \right) y_{nj} - \mu_j^{(3.39)}$$

$$\text{subject to:} \quad \sum_{n \in N} r_n y_{nj} \leq R_j \tag{3.45}$$

$$y_{nj} \in \{0,1\} \qquad n \in N, \tag{3.46}$$

where y_{nj} are the binary decision variables equal to 1 if and only if destination n is assigned to outbound door j. Similarly to PP_i^X, the objective function implicitly evaluates the reduced cost of all feasible configurations in H_j. The optimal solution to PP_j^Y is thus an outbound door configuration for j having the smallest reduced cost coefficient. PP_j^Y also corresponds to a 0-1 knapsack problem ans can be solved in the same way as PP_i^X.

Finally, the third pricing problem is associated with the routing variables z_{kij}. It can also be decomposed into $|K|$ independent subproblems which in turn, can be solved by inspection. In particular, for each commodity k, we simply identify the door pair $(i,j) \in I \times J$ with the smallest reduced cost coefficient:

$$[PP_k^Z] \quad (i_k, j_k) \in \arg\min \left\{ (i,j) \in I \times J : \sum_{i\in I}\sum_{j \in J} w_k d_{ij} - \mu_{ki}^{(3.34)} - \mu_{kj}^{(3.35)} \right.$$
$$\left. -r_{d(k)}\mu_{mij}^{(3.37)} - s_{o(k)}\mu_{nji}^{(3.36)} \right\} - \mu_k^{(3.33)}$$

This leads to a $O(|K||I||J|)$ complexity for solving all PP_k^Z pricing problems.

3.5.3 Initial Columns

Initial columns are generated by solving two independent Generalized Assignment Problems (GAPs). Each GAP provides feasible assignments on an inbound (outbound) side of the cross-dock. Whenever the assignments, x_{mi} and y_{nj} are generated, we use that information to calculate the path based variables as $p_{kij} = x_{o(k)i}y_{d(k)j}$. Moreover, the algorithm starts with a few z_{kij} variables that satisfy: $|i - j| \leq 1$. The rest of these variables is generated using the third pricing problem.

3.5.4 Termination Criterion

A valid lower bound $v(LB)$ on the master problem MP can be obtained as seen in Amor et al. [6], and then be used in a termination criterion. Such a valid lower bound is calculated iteratively via solving relaxations of the RMP. At each CG iteration, whenever we price out new variables or columns from the pricing problems, PP_i^X, PP_j^Y and PP_k^Z, their incorporation will decrease the current optimal value associated with the RMP, i.e., $v(RMP^t)$. For simplicity, we restate PP_i^X, PP_j^Y and PP_k^Z as PP_i, PP_j and PP_k, respectively. The valid lower bound for a given iteration t can be stated as follows:

$$v(LB^t) = v(RMP^t) + \sum_{i\in I} v(PP_i^t) - \mu_i^{(3.38)} + \sum_{j \in J} v(PP_j^t) - \mu_j^{(3.39)} + \sum_{k\in K} v(PP_k^t) - \mu_k^{(3.33)}$$

The dual variables $\mu_i^{(3.38)}$, $\mu_j^{(3.39)}$ and $\mu_k^{(3.33)}$ are associated with the convexity constraints (3.38), (3.39) and (3.33), respectively. Hence, they are singled out.
Finally, we use this computation to terminate CG whenever: $v(LB^t) - v(RMP^t) \leq 10^{-6}$

3.6 Computational Experiments

In this section, we present the results of computational experiments performed to assess the behavior of the MIP formulations, combinatorial, linear and Lagrangean relaxations for the CDAP.

All algorithms were coded in C and run on an Intel Xeon E3 1240 V2 processor with 3.40 GHz and 24GB of RAM memory under a Windows environment. The MIP formulations and CG algorithm were implemented using the callable library of CPLEX 12.6.2. We first explain the data set generation, then the tuning of the parameters and finally, the computational comparisons and analysis.

3.6.1 Data Set Generation

We use the benchmark data set introduced by Guignard et al. [28] for the CDAP with only transfer times, and then, we extend it to include unloading/loading times.

Guignard et al. [28] generate the flow matrix with integer numbers between $[10, 50]$ until 25% of the matrix is full. The distance (or travel time) matrix is generated within the interval $[\tau, \tau + I - 1]$, where $\tau = 8$ indicates the distance between two doors facing each other, and then an increment of 1 unit is added for the next closest door. All instances are generated with $|I| = |J|$ for a rectangular shaped cross-dock. Capacities are identical and calculated by dividing the total flow coming from all origins by the total number of inbound doors, and then adding some capacity slackness. A total of 50 instances are considered from the possible combinations of these parameters and are denoted as 00x00S00. The first position refers to the number of origins/destinations, taken in $\{8, 9, 10, 11, 12, 15, 20\}$. The second position refers to the number of inbound/outbound doors, taken in $\{4, 5, 6, 7, 10\}$. The third position refers to the capacity slackness which are drawn from $\{5\%, 10\%, 15\%, \%20, \%30\}$.

We generate the additional parameters of unloading/loading times as follows. Using a private communication with a large cross-dock company[1], and based on their data, we find out that it is generally assumed that the total handling time of shipments inside the cross-dock is divided approximately into 22% for unloading, %43 for traveling, and 35% for loading. We observe that traveling time takes the largest portion of handling time inside the cross-dock, while loading is 1.5 to 2 times longer than unloading as mentioned in Zhang et al. [61]. As mentioned in Bartholdi and Hackman [9], loading is more difficult than unloading as the loader employee has to make sure the truck is fully packed and that usually requires double handling. Therefore, using a uniform distribution, we generate the unloading and loading times with the above distribution with respect

[1]The company does not want to be identified

to the previously generated traveling time matrix. That is for $i \in I$, $u_i = 50\%\textsc{random}[\tau, \tau+I-3]$ where about $50\% \approx \frac{22}{43}100$, and for $j \in J$, $\ell_j = 80\%\textsc{random}[\tau, \tau + I - 3]$ where $80\% \approx \frac{35}{43}100$. The expression $\tau+I-3$ was obtained after some fine tuning so that the 20%, 34%, 35% percentages are observed at optimal solution values.

3.6.2 Tuning and Termination Criteria of the Algorithms

The following parameters were fine tuned with preliminary experiments so as to offer the best compromise between the quality of the solutions and computational times. For the MIP formulations, we use the deterministic parallel branch-and-cut algorithm of CPLEX with 6 threads and the central processing unit (CPU) time limit was set to 7,200 seconds (i.e., 2 hours). The remaining CPLEX parameters were kept to their default value. For the standard CDAP, the sub-gradient method was implemented as explained in Nassief et al. [42], reporting only lower bounds and CPU times. For the CDAP with unloading/loading times, the LR was adapted to incorporate the additional linear terms and the subgradient method was also used to solve the associated Lagrangean dual problem. The maximum number of iterations was set to 300,000, the step length parameter ε was multiplied by 0.9 every 2,000 consecutive iterations without improvement of the lower bound and reset to 2 whenever $\varepsilon < 10^{-7}$.

3.6.3 Comparisons of Formulations and Relaxations

In the following experiments, we compare the results obtained with the MIP formulations, the combinatorial relaxation, and lower bounds obtained by CG and LR. The detailed and summarized results of these comparisons are given in Tables 3.1 and 3.2, respectively, for the CDAP with unloading/loading times, and in Tables 3.3 and 3.4, respectively, for the standard CDAP. In both tables, the first column gives the name of the instance. Under each formulation, we report its %LP deviation with the respect to the optimal solution, the remaining optimality %gap when terminating, the number of explored branch and bound (B&B) nodes, and the CPU time in seconds. The LP deviation is calculated as $\%LP = 100(OPT - LP)/(OPT)$, where OPT is the optimal value and LP is the optimal solution for the LP relaxation of each formulation. For $P1$ and $P2$, we show the LP deviation with and without the valid inequalities (3.13) and (3.14), that is $\%LP'$ and $\%LP$, respectively. The combinatorial bounds are obtained in less than a second for all instances, and so we report only on the quality of their bounds. For the LR, we report the deviation of the best obtained lower bound and CPU time after termination. For the DC' formulation, we report its %LP using CG algorithm and the CPU time. Finally, we use $P1$ and $P2$ instead of $P1'$ and $P2'$ to obtain optimal solutions in CPLEX as the former outperform the latter with respect to the required CPU time and remaining %gap for unsolved instances.

Table 3.1: Computational Comparisons - CDAP

Instances	Path-based (P1)					Path-based (P2)					Flow-based (F1)				COMB	LR'		DC'	
	%LP	%LP'	%gap	B&B	CPU	%LP	%LP'	%gap	B&B	CPU	%LP	%gap	B&B	CPU	%LB	%LB	CPU	%LB	CPU
8x4S5	3.12	1.56	0.00	42	0.20	3.12	1.56	0.00	0	0.07	3.12	0.00	0	0.07	5.20	1.50	28.58	1.14	0.15
8x4S10	2.95	1.53	0.00	100	0.10	2.95	1.53	0.00	89	0.13	2.95	0.00	172	0.14	4.08	3.22	24.76	1.32	0.14
8x4S15	3.11	1.73	0.00	63	0.11	3.11	1.73	0.00	0	0.09	3.11	0.00	0	0.11	5.92	4.86	27.34	1.25	0.14
8x4S20	2.77	1.27	0.00	0	0.07	2.77	1.27	0.00	0	0.12	2.77	0.00	0	0.12	4.68	7.15	26.05	0.87	0.09
8x4S30	2.39	1.17	0.00	127	0.21	2.39	1.17	0.00	74	0.11	2.39	0.00	0	0.11	6.04	11.30	26.46	1.05	0.11
9x4S5	3.36	1.65	0.00	187	0.17	3.36	1.65	0.00	116	0.12	3.36	0.00	0	0.09	8.14	2.32	28.39	1.45	0.16
9x4S10	3.35	1.63	0.00	126	0.13	3.35	1.63	0.00	156	0.09	3.35	0.00	0	0.23	7.40	4.36	27.91	1.52	0.12
9x4S15	3.33	1.62	0.00	107	0.09	3.33	1.62	0.00	64	0.08	3.33	0.00	66	0.10	5.71	6.90	28.29	1.36	0.11
9x4S20	3.13	1.58	0.00	57	0.09	3.13	1.58	0.00	0	0.04	3.13	0.00	128	0.11	3.53	9.78	29.58	1.27	0.11
9x4S30	3.34	1.52	0.00	100	0.11	3.34	1.52	0.00	92	0.08	3.34	0.00	114	0.10	4.85	14.30	30.83	1.48	0.12
10x4S5	3.92	2.89	0.00	399	0.22	3.92	2.89	0.00	417	0.26	3.92	0.00	1,411	0.26	5.14	4.15	33.10	2.82	0.18
10x4S10	2.52	1.59	0.00	219	0.14	2.52	1.59	0.00	160	0.12	2.52	0.00	350	0.16	5.77	4.28	34.25	1.58	0.17
10x4S15	2.53	1.60	0.00	133	0.15	2.53	1.60	0.00	249	0.20	2.53	0.00	327	0.13	6.49	7.28	37.56	1.54	0.13
10x4S20	2.54	1.64	0.00	276	0.13	2.54	1.64	0.00	185	0.13	2.54	0.00	405	0.16	5.66	9.75	33.89	1.60	0.16
10x4S30	1.50	0.91	0.00	0	0.06	1.50	0.91	0.00	52	0.08	1.50	0.00	161	0.18	5.12	13.13	36.54	0.88	0.15
10x5S5	4.24	2.87	0.00	750	0.45	4.24	2.87	0.00	519	0.39	4.24	0.00	1,304	0.78	11.13	3.14	39.64	2.60	0.19
10x5S10	3.54	2.29	0.00	638	0.31	3.54	2.29	0.00	619	0.28	3.54	0.00	1,414	0.54	8.84	4.89	40.67	2.06	0.18
10x5S15	3.04	1.92	0.00	502	0.36	3.04	1.92	0.00	508	0.28	3.04	0.00	873	0.47	10.56	5.07	42.11	1.77	0.21
10x5S20	2.97	1.88	0.00	194	0.29	2.97	1.88	0.00	183	0.21	2.97	0.00	648	0.39	9.58	7.20	42.83	1.43	0.22
10x5S30	3.15	2.21	0.00	460	0.30	3.15	2.21	0.00	530	0.27	3.15	0.00	1,531	0.53	8.94	10.90	42.16	1.95	0.23
11x5S5	4.88	3.37	0.00	1,911	0.88	4.88	3.37	0.00	2,162	0.75	4.88	0.00	1,701	0.95	8.60	4.02	46.88	3.11	0.35
11x5S10	3.59	2.07	0.00	552	0.37	3.59	2.07	0.00	822	0.41	3.59	0.00	3,458	1.22	10.45	5.00	47.92	1.91	0.32
11x5S15	3.36	2.13	0.00	436	0.36	3.36	2.13	0.00	760	0.42	3.36	0.00	3,308	1.26	9.07	7.30	47.11	1.99	0.23
11x5S20	3.21	2.05	0.00	394	0.28	3.21	2.05	0.00	387	0.25	3.21	0.00	3,360	1.41	10.90	8.11	49.84	1.83	0.28
11x5S30	2.87	1.83	0.00	459	0.33	2.87	1.83	0.00	302	0.44	2.87	0.00	1,404	0.57	6.37	12.33	59.85	1.77	0.28

The CDAP with Unloading/Loading Times

In terms of MIP formulations, the computational experiments in Tables 3.1 and 3.2 show that both path-based formulations $P1$ and $P2$, and the flow-based formulation $F1$ are able to obtain the optimal solutions of 45 out of 50 instances within the time limit of 2 hours. For the remaining

Table 3.2: Computational Comparisons - CDAP

Instances	Path-based (P1)					Path-based (P2)					Flow-based (F1)				COMB	LR'		DC'	
	%LP	%LP'	%gap	B&B	CPU	%LP	%LP'	%gap	B&B	CPU	%LP	%gap	B&B	CPU	%LB	%LB	CPU	%LB	CPU
12x5S5	3.81	2.53	0.00	876	0.81	3.81	2.53	0.00	711	0.58	3.81	0.00	1,084	0.80	7.97	3.26	65.46	2.12	0.40
12x5S10	3.15	2.02	0.00	878	0.62	3.15	2.02	0.00	956	0.61	3.15	0.00	3,027	1.31	7.26	5.44	65.96	1.96	0.34
12x5S15	3.14	2.10	0.00	697	0.57	3.14	2.10	0.00	770	0.63	3.14	0.00	1,754	0.87	7.46	6.74	66.51	1.98	0.33
12x5S20	3.20	2.28	0.00	1,301	0.78	3.20	2.28	0.00	1,369	0.65	3.20	0.00	5,585	2.04	7.46	8.33	70.18	2.14	0.33
12x5S30	3.18	2.36	0.00	4,874	2.52	3.18	2.36	0.00	2,989	1.34	3.18	0.00	3,076	1.33	6.67	12.56	75.20	2.28	0.36
12x6S5	5.92	4.50	0.00	5,298	4.19	5.92	4.50	0.00	3,863	2.86	5.92	0.00	2,525	2.09	12.09	4.54	72.47	4.03	0.97
12x6S10	4.36	2.82	0.00	1,704	1.55	4.36	2.82	0.00	2,865	1.75	4.36	0.00	3,227	2.11	10.15	4.97	75.19	2.48	0.64
12x6S15	4.46	3.02	0.00	1,845	1.51	4.46	3.02	0.00	1,883	1.33	4.46	0.00	2,844	2.25	8.49	7.14	74.56	2.63	0.61
12x6S20	3.93	2.58	0.00	3,643	2.36	3.93	2.58	0.00	3,993	2.21	3.93	0.00	4,441	3.02	12.66	8.17	71.36	2.36	0.50
12x6S30	3.83	2.70	0.00	5,288	2.59	3.83	2.70	0.00	2,902	1.62	3.83	0.00	7,012	3.51	8.73	11.75	70.89	2.48	0.54
15x6S5	4.18	2.57	0.00	24,105	29.36	4.18	2.57	0.00	18,627	18.69	4.18	0.00	57,844	52.27	15.17	4.24	99.75	2.53	1.79
15x6S10	4.10	2.53	0.00	9,397	10.30	4.10	2.53	0.00	9,903	8.98	4.10	0.00	94,988	61.39	10.14	6.51	98.04	2.48	1.70
15x6S15	3.86	2.52	0.00	19,195	20.59	3.86	2.52	0.00	18,192	16.48	3.86	0.00	135,875	81.87	12.12	7.61	100.94	2.48	1.51
15x6S20	3.92	2.66	0.00	38,414	33.57	3.92	2.66	0.00	22,273	16.87	3.92	0.00	134,754	75.27	8.31	9.72	101.89	2.63	1.52
15x6S30	3.45	2.21	0.00	13,203	8.61	3.45	2.21	0.00	10,972	6.43	3.45	0.00	99,867	64.19	12.16	12.91	105.19	2.20	1.44
15x7S5	5.77	3.58	0.00	29,282	67.59	5.77	3.58	0.00	47,041	69.74	5.77	0.00	274,350	249.63	19.44	4.34	110.26	3.40	2.76
15x7S10	5.51	3.33	0.00	31,911	52.83	5.51	3.33	0.00	19,626	29.84	5.51	0.00	481,906	387.24	17.41	5.29	115.07	3.03	3.17
15x7S15	5.96	3.56	0.00	16,870	27.98	5.96	3.56	0.00	18,970	24.99	5.96	0.00	275,122	244.73	13.88	7.19	114.69	3.26	3.14
15x7S20	5.23	3.30	0.00	35,980	59.75	5.23	3.30	0.00	40,073	58.00	5.23	0.00	361,863	300.70	10.42	9.09	115.56	3.06	2.65
15x7S30	4.21	2.62	0.00	61,949	72.24	4.21	2.62	0.00	63,788	60.13	4.21	0.00	454,743	337.91	12.11	10.25	122.10	2.51	2.14
20x10S5	7.96	5.41	2.19	329,564	time	7.96	5.41	2.56	457,147	time	7.96	2.57	982,088	time	19.37	6.19	288.61	4.99	39.59
20x10S10	6.81	4.63	2.15	411,539	time	6.81	4.63	1.51	438,959	time	6.81	2.14	1,054,134	time	25.31	6.00	285.62	3.98	41.84
20x10S15	6.90	4.77	3.29	455,900	time	6.90	4.77	3.56	521,212	time	6.90	3.74	1,184,303	time	22.13	6.72	293.04	4.43	38.33
20x10S20	7.04	5.00	3.65	460,859	time	7.04	5.00	2.84	473,477	time	7.04	3.67	1,276,100	time	19.01	8.28	293.18	4.64	35.50
20x10S30	7.47	5.05	2.83	474,231	time	7.47	5.05	3.15	547,790	time	7.47	3.45	1,416,406	time	19.90	9.44	303.20	4.78	34.36
Average	4.00	2.54	0.28	48,941	728.12	4.00	2.54	0.27	54,776	726.58	4.00	0.31	166,821	757.69	10.08	7.18	83.35	2.33	4.42

instances where the optimal solution is not found within 2 hours, the tightest remaining average %gap of 2.72 is obtained by $P2$ as opposed to 2.82 and 3.11 in $P1$ and $F1$, respectively. While both path-based formulations have the same theoretical bounds in terms of their LP relaxations, $P2$, on average, slightly outperforms $P1$ in terms of required CPU time in the largest set of instances.

In terms of relaxations, we demonstrate in this set of instances the dominance of this new configuration-based formulation DC' over all other formulations, $P2$, and $F1$ that we introduce in this paper, as well as the $P1$ of Nassief et al. [42]. We highlight that while both DC' and LR have the same theoretical bounds as shown in Section 3.4, the CG algorithm obtains strictly better lower bounds as compared to LR given the slow convergence of the subgradient method. Moreover, as the capacities looses for each block of instances, CG maintains its dominance by far as opposed to the subgradient method.

The Standard CDAP

In terms of MIP formulations, the computational experiments in Tables 3.3 and 3.4 show also that both path-based formulations $P1$ and $P2$, and the flow-based formulation $F1$ are able to obtain the optimal solutions of 45 out of 50 instances within the time limit of 2 hours. For the remaining instances where the optimal solution is not found within 2 hours, the tightest remaining average %gap of 4.68, this time, is obtained by $P1$ as opposed to 5.58 and 7.74 in $P2$ and $F1$, respectively. Contrary to the CDAP with unloading/loading times, formulation $P1$ seems to be slightly better than $P2$ in terms of the CPU time and remaining %gap.

In terms of relaxations, we demonstrate in this set of instances, again, the dominance of this new configuration-based formulation DC' over all other formulations. We highlight that while both DC' and LR have the same theoretical bounds as shown in Section 3.4, CG algorithm outperforms the LR in terms of the quality of the bounds as well as the CPU time. However, we observe from Tables 3.3 and 3.4 that the subgradient method has a more stable convergence regardless of the capacity slackness, as opposed to the case in Tables 3.1 and 3.2. Moreover, we notice that for the standard CDAP as seen in Tables 3.3 and 3.4, all %LP of $P1$, $P2$, and $F1$ coincide with the combinatorial bounds even though theoretically they are better.

3.6.4 Sensitivity Analysis

In these experiments, we would like to highlight the impact that the parameter τ used in the

Table 3.3: Computational Comparisons - Standard CDAP

Instances	Path-based (P1)					Path-based (P2)					Flow-based (F1)				COMB	LR'		DC'	
	%LP	%LP'	%gap	B&B	CPU	%LP	%LP'	%gap	B&B	CPU	%LP	%gap	B&B	CPU	%LB	%LB	CPU	%LB	CPU
8x4S5	6.76	3.26	0.00	0	0.08	6.76	3.26	0.00	0	0.07	6.76	0.00	0	0.06	6.76	3.51	4.62	2.80	0.20
8x4S10	6.67	3.39	0.00	45	0.12	6.67	3.39	0.00	0	0.10	6.67	0.00	229	0.26	6.67	3.63	5.19	3.10	0.16
8x4S15	5.63	2.53	0.00	0	0.08	5.63	2.53	0.00	0	0.13	5.63	0.00	0	0.05	5.63	2.87	5.38	2.24	0.11
8x4S20	5.15	2.24	0.00	0	0.18	5.15	2.24	0.00	19	0.08	5.15	0.00	0	0.14	5.15	2.49	5.47	1.97	0.09
8x4S30	4.72	2.16	0.00	61	0.08	4.72	2.16	0.00	48	0.06	4.72	0.00	0	0.11	4.72	3.62	7.12	2.16	0.07
9x4S5	7.66	3.71	0.00	100	0.13	7.66	3.71	0.00	0	0.08	7.66	0.00	0	0.18	7.66	4.19	6.45	3.44	0.14
9x4S10	7.35	3.69	0.00	77	0.09	7.35	3.69	0.00	94	0.12	7.35	0.00	235	0.09	7.35	4.17	6.16	3.47	0.13
9x4S15	6.56	3.16	0.00	75	0.21	6.56	3.16	0.00	0	0.13	6.56	0.00	0	0.05	6.56	3.54	14.04	3.05	0.14
9x4S20	5.95	2.75	0.00	0	0.09	5.95	2.75	0.00	58	0.09	5.95	0.00	132	0.08	5.95	3.34	14.69	2.72	0.10
9x4S30	5.42	2.63	0.00	0	0.04	5.42	2.63	0.00	75	0.09	5.42	0.00	222	0.08	5.42	4.41	11.32	2.63	0.07
10x4S5	8.56	6.31	0.00	384	0.16	8.56	6.31	0.00	304	0.16	8.56	0.00	1,176	0.30	8.56	6.69	9.90	6.27	0.15
10x4S10	5.77	3.67	0.00	253	0.18	5.77	3.67	0.00	179	0.12	5.77	0.00	0	0.09	5.77	4.62	10.16	3.67	0.12
10x4S15	5.34	3.45	0.00	164	0.12	5.34	3.45	0.00	71	0.10	5.34	0.00	856	0.21	5.34	4.56	12.17	3.45	0.10
10x4S20	4.90	3.19	0.00	230	0.10	4.90	3.19	0.00	131	0.18	4.90	0.00	763	0.17	4.90	4.49	12.61	3.19	0.11
10x4S30	3.76	2.35	0.00	0	0.06	3.76	2.35	0.00	0	0.06	3.76	0.00	417	0.11	3.76	3.66	13.40	2.35	0.08
10x5S5	9.79	6.48	0.00	521	0.29	9.79	6.48	0.00	510	0.29	9.79	0.00	1,242	0.67	9.79	7.20	9.24	6.32	0.17
10x5S10	7.84	4.78	0.00	515	0.24	7.84	4.78	0.00	387	0.25	7.84	0.00	1,377	0.49	7.84	5.71	11.96	4.66	0.14
10x5S15	6.71	3.87	0.00	346	0.26	6.71	3.87	0.00	334	0.22	6.71	0.00	1,530	0.46	6.71	5.18	12.18	3.76	0.15
10x5S20	5.90	3.31	0.00	77	0.19	5.90	3.31	0.00	164	0.18	5.90	0.00	1,382	0.52	5.90	4.56	12.86	3.18	0.13
10x5S30	5.39	3.17	0.00	350	0.20	5.39	3.17	0.00	414	0.20	5.39	0.00	1,937	0.59	5.39	4.32	12.69	3.17	0.17
11x5S5	10.80	7.33	0.00	1,440	0.68	10.80	7.33	0.00	1,657	0.54	10.80	0.00	2,613	1.15	10.80	8.36	14.91	6.96	0.22
11x5S10	7.98	4.77	0.00	840	0.35	7.98	4.77	0.00	663	0.36	7.98	0.00	4,156	1.17	7.98	6.02	16.51	4.68	0.22
11x5S15	7.52	4.59	0.00	529	0.28	7.52	4.59	0.00	493	0.23	7.52	0.00	5,388	1.60	7.52	5.97	14.09	4.55	0.20
11x5S20	6.33	3.61	0.00	226	0.23	6.33	3.61	0.00	82	0.27	6.33	0.00	3,435	1.04	6.33	5.00	17.60	3.59	0.19
11x5S30	6.09	3.76	0.00	251	0.22	6.09	3.76	0.00	334	0.23	6.09	0.00	4,462	1.40	6.09	5.32	19.48	3.76	0.21

Table 3.4: Computational Comparisons - Standard CDAP

Instances	Path-based (P1)					Path-based (P2)					Flow-based (F1)				COMB	LR'		DC'	
	%LP	%LP'	%gap	B&B	CPU	%LP	%LP'	%gap	B&B	CPU	%LP	%gap	B&B	CPU	%LB	%LB	CPU	%LB	CPU
12x5S5	8.23	5.28	0.00	601	0.60	8.23	5.28	0.00	494	0.49	8.23	0.00	2,133	0.89	8.23	6.06	12.24	5.19	0.36
12x5S10	7.14	4.45	0.00	296	0.33	7.14	4.45	0.00	395	0.34	7.14	0.00	3,425	1.20	7.14	5.58	15.62	4.41	0.30
12x5S15	6.69	4.21	0.00	379	0.42	6.69	4.21	0.00	352	0.36	6.69	0.00	2,122	0.58	6.69	5.44	15.31	4.17	0.27
12x5S20	6.69	4.46	0.00	708	0.49	6.69	4.46	0.00	834	0.47	6.69	0.00	3,451	0.97	6.69	5.77	18.90	4.42	0.28
12x5S30	6.50	4.61	0.00	1,437	0.64	6.50	4.61	0.00	1,468	0.64	6.50	0.00	6,455	2.15	6.50	6.29	17.26	4.61	0.24
12x6S5	13.03	9.53	0.00	4,663	3.47	13.03	9.53	0.00	5,553	3.41	13.03	0.00	15,850	8.26	13.03	10.16	15.63	9.34	0.62
12x6S10	9.41	6.06	0.00	1,395	1.01	9.41	6.06	0.00	1,214	0.88	9.41	0.00	9,452	5.04	9.41	6.57	17.72	5.99	0.44
12x6S15	8.59	5.42	0.00	1,461	1.18	8.59	5.42	0.00	873	0.89	8.59	0.00	6,413	3.25	8.59	6.36	19.65	5.41	0.47
12x6S20	8.15	5.19	0.00	1,522	1.28	8.15	5.19	0.00	1,073	0.85	8.15	0.00	15,478	7.06	8.15	6.33	23.16	5.18	0.38
12x6S30	7.39	4.82	0.00	2,120	1.20	7.39	4.82	0.00	986	0.60	7.39	0.00	10,558	4.39	7.39	6.51	29.26	4.81	0.43
15x6S5	9.93	6.14	0.00	7,304	7.85	9.93	6.14	0.00	8,525	7.03	9.93	0.00	116,161	83.31	9.93	7.04	30.87	6.10	1.73
15x6S10	9.12	5.65	0.00	7,375	7.30	9.12	5.65	0.00	4,628	4.41	9.12	0.00	232,918	135.83	9.12	6.66	32.87	5.64	1.49
15x6S15	8.87	5.66	0.00	11,507	11.25	8.87	5.66	0.00	5,777	5.30	8.87	0.00	175,697	103.78	8.87	6.97	35.63	5.66	1.37
15x6S20	8.57	5.61	0.00	9,956	8.81	8.57	5.61	0.00	10,384	7.54	8.57	0.00	219,771	109.26	8.57	7.19	29.57	5.61	1.40
15x6S30	7.54	4.98	0.00	5,262	4.91	7.54	4.98	0.00	9,127	5.93	7.54	0.00	154,866	73.06	7.54	6.77	31.36	4.98	1.37
15x7S5	13.70	8.51	0.00	25,022	43.08	13.70	8.51	0.00	61,186	84.45	13.70	0.00	411,518	350.38	13.70	9.51	21.54	8.30	2.49
15x7S10	12.28	7.44	0.00	23,452	34.56	12.28	7.44	0.00	24,173	28.17	12.28	0.00	1,719,101	1003.13	12.28	8.62	24.28	7.35	2.45
15x7S15	11.36	6.80	0.00	17,484	20.57	11.36	6.80	0.00	14,603	17.77	11.36	0.00	1,700,274	1003.46	11.36	8.13	25.45	6.77	2.30
15x7S20	10.49	6.19	0.00	11,166	13.35	10.49	6.19	0.00	7,781	9.00	10.49	0.00	603,824	396.60	10.49	7.81	34.73	6.17	2.00
15x7S30	9.83	6.06	0.00	11,864	15.06	9.83	6.06	0.00	16,646	16.63	9.83	0.00	597,416	369.92	9.83	7.90	42.17	6.05	1.95
20x10S5	17.98	12.16	2.82	429,752	time	17.98	12.16	6.71	522,011	time	17.98	7.39	986,768	time	17.98	12.96	63.76	11.99	37.15
20x10S10	16.02	10.52	2.69	476,365	time	16.02	10.52	2.04	494,121	time	16.02	6.56	1,185,087	time	16.02	11.43	69.21	10.31	32.29
20x10S15	15.73	10.61	7.23	440,472	time	15.73	10.61	5.65	419,329	time	15.73	7.21	1,428,785	time	15.73	11.13	79.04	10.53	33.95
20x10S20	15.18	10.41	5.61	407,833	time	15.18	10.41	7.65	446,603	time	15.18	9.30	1,409,018	time	15.18	11.76	82.69	10.36	30.97
20x10S30	13.98	9.73	5.07	504,245	time	13.98	9.73	5.87	497,148	time	13.98	8.22	1,846,914	time	13.98	11.35	98.46	9.71	29.90
Average	8.54	5.29	0.47	48,203	723.64	8.54	5.29	0.56	51,226	723.99	8.54	0.77	257,900	793.47	8.54	6.35	23.33	5.20	3.80

generation of travel times (or distances) t_{ij} has in the perceived quality of the obtained LP bounds. We note that due to the fact that τ appears in every single t_{ij} and that each commodity is routed exactly via one pair of doors (i, j), the constant term $\tau|K|$ can be removed from the objective function. That is, the set of optimal solutions remain the same regardless of the τ value used in the experiments. As mentioned earlier, this data was generated on the assumption that a direct distance between two doors is assumed $\tau = 8$ as in Guignard et al. [28]. However, changing this parameter and updating the matrix accordingly, will provide different %LP deviations while maintaining the same difficulty for solving CDAP. In Tables 3.5 and 3.6, we assign different values to τ in the interval $[0, 1, 000]$ using P1 and P1' for the both CDAPs.

Whenever $\tau = 8$, the %LPs clearly match the ones we reported in previous tables. However, these deviations significantly decrease(increase) as τ increases(decrease). It is thus important to keep in mind the impact of the parameter τ when computing the %LP deviations and using them to determine how *tight* or good an MIP formulation might be for the CDAPs.

Table 3.5: Impact of Travel Times on %LP - CDAP

	%LP with different τ values							%LP' with different τ values						
instances	0	1	8	20	50	100	1000	0	1	8	20	50	100	1000
8x4S5	60.05	18.32	3.12	1.29	0.52	0.26	0.03	30.05	9.17	1.56	0.65	0.26	0.13	0.01
8x4S10	36.59	15.10	2.95	1.24	0.51	0.26	0.03	19.00	7.84	1.53	0.64	0.26	0.13	0.01
8x4S15	52.49	17.57	3.11	1.29	0.52	0.26	0.03	29.18	9.77	1.73	0.72	0.29	0.15	0.01
8x4S20	59.10	16.66	2.77	1.14	0.46	0.23	0.02	27.16	7.66	1.27	0.52	0.21	0.11	0.01
8x4S30	39.55	13.43	2.39	0.99	0.40	0.20	0.02	19.43	6.60	1.17	0.49	0.20	0.10	0.01
Average	49.56	16.22	2.87	1.19	0.48	0.24	0.03	24.96	8.21	1.45	0.60	0.24	0.12	0.01

Table 3.6: Impact of Travel Times on %LP - Standard CDAP

	%LP with different τ values							%LP' with different τ values						
instances	0	1	8	20	50	100	1000	0	1	8	20	50	100	1000
8x4S5	100	36.73	6.76	2.82	1.15	0.58	0.06	48.16	17.69	3.26	1.36	0.55	0.28	0.03
8x4S10	100	36.39	6.67	2.78	1.13	0.57	0.06	50.81	18.49	3.39	1.41	0.57	0.29	0.03
8x4S15	100	32.32	5.63	2.33	0.95	0.48	0.05	44.95	14.53	2.53	1.05	0.43	0.21	0.02
8x4S20	100	30.29	5.15	2.13	0.86	0.43	0.04	43.39	13.14	2.24	0.92	0.37	0.19	0.02
8x4S30	100	28.38	4.72	1.94	0.79	0.39	0.04	45.78	12.99	2.16	0.89	0.36	0.18	0.02
Average	100	32.82	5.79	2.40	0.98	0.49	0.05	46.62	15.37	2.72	1.13	0.46	0.23	0.02

3.7 Conclusions

In this paper we studied cross-dock door assignment problems with and without loading/unloading times. We presented two new mathematical programming formulations which were analytically and computationally compared with existing ones. In particular, we compared them with respect to the quality of their linear programming relaxation bounds and with respect to Lagrangean bounds presented in Nassief et al. [42]. We showed than the LP relaxation of the configuration based formulation is equivalent to the Lagrangean Dual problem of the LR given in Nassief et al. [42]. However, the results of computational experiments indicate that, when using a column generation algorithm instead of a subgradient optimization algorithm, better bounds can be obtained in practice. This behavior is mainly attributed to the slow convergence of the subgradient algorithm as compared to a CG for this particular class of problems. We also pointed out the impact of parameters used in the generation of travel times (or distances) between doors and the perceived quality of the LP bounds obtained with the formulations.

Chapter 4

The Container Scheduling and Cross-dock Door Selection Problem

In this paper, we introduce an integrated scheduling and selection problem that is motivated by a cross-docking application. The daily decisions of scheduling containers and selecting dock doors to unload these containers are carried out simultaneously. The objective is to minimize the total weighted tardiness cost resulting from scheduling containers plus the total labor cost resulting from selecting dock doors to unload containers. We introduce two integer programming models for static and dynamic environments. Computational experiments show that our new static model significantly outperforms the best existing one, and the dynamic one is able to solve real life instances optimally in a reasonable time.

The following paper was submitted to the *International Journal of Production Research* in March 2017. Nassief et al. [43].

4.1 Introduction

Cross-docking now plays a vital distribution role in supply chain networks around the world. It is a modern logistics strategy that has been increasingly and successfully adopted by companies in North America, Europe and Asia. Due to the increasing demand for a wide variety of manufactured goods, *storage* and *order picking* functions of warehouses are nearly eliminated in the delivery chain. For examples of successful cross-docking implementations, the interested reader is referred to Forger [25], Kinnear [31], Witt [60], Chen and Song [16], and Napolitano [41].

Cross-docks are designed to facilitate the rapid transshipment of highly and consistently demanded materials, resulting in better service level and quicker response across the supply chain at reduced cost. On a daily basis, incoming and outgoing trucks arrive at cross-dock facilities and are intelligently assigned to doors. Goods are unloaded from the incoming vehicles, consolidated in a staging area according to their destinations, and then loaded onto outgoing trailers, with only a few hours storage in between. Daily operational decisions are made on the fly. This includes assigning and sequencing incoming and outgoing trailers over doors, selecting dock doors for unloading/loading, assigning workforces, and even routing trucks that pick up and deliver the vast array of goods. While each daily operational decision plays an important managerial role independently, the interaction between these decisions is crucial to effectively coordinate the incoming and outgoing flows. Assigning trucks to dock doors is a classical cross-dock door assignment problem, and is usually carried out over a midterm planning horizon where the origins and destinations of the trucks are assigned to inbound and outbound doors, respectively, as seen in Zhu et al. [62], Guignard et al. [28] and Nassief et al. [42]. However, in a short term planning horizon, there are more incoming and outgoing trucks than the number of available doors, and hence, sequencing and assignment decisions are integrated to provide daily schedules, as seen in Miao et al. [40], Boysen et al. [11] and Gelareh et al. [26]. The selection of dock doors to unload or load trucks in a given day or shift has an impact on workforce (i.e., labor cost), and has also been integrated with other decisions such as the assignment of trucks to dock doors as seen in Rosales et al. [48]. Other integrated decisions such as the assignment and routing of trucks are reported in the literature as seen in Enderer et al. [23]. The more these interdependent decisions are made simultaneously, the more added value is achieved in terms of minimizing the total operational cost. This may increase the complexity of modeling, so care should be taken to avoid optimizations goals that would result in little benefit.

In this paper, we start by presenting a fundamental container scheduling problem that involves the assignment and sequencing of arriving containers over inbound dock doors with the objective of minimizing the total weighted tardiness. Then, we introduce a new integrated cross-docking problem of scheduling incoming containers and selecting dock doors to unload these containers in order to minimize the total weighted tardiness and labor costs. Both problems, scheduling and the integration of dock door selection, are driven from our observations on a cross-dock facility in the USA. We study this problem and provide models for both static and dynamic environments where the decisions maker is able to plan over several shifts/days.

Two to three times a week, vessels carrying shipping containers from different overseas manufacturers arrive at a coastal port. On weekdays, tractor trucks arrive at the port, pick up a certain amount of containers and deliver them to the cross-dock facility's yard, and return other empty ones (i.e., empties) to the port. Each container must be transported to the cross-dock facility, parked in its receiving yard, scheduled to arrive at a selected strip door, unloaded by one worker, then after being emptied, returned to the port before its due date to avoid penalty.

The quantity of arriving full containers at the port terminal ranges from hundreds to thousands a week, while the quantity of containers arriving at the receiving yard on a given day are about a hundred. The cross-dock facility has an I-shaped layout with a 150 dock doors for unloading/loading. The company can operate a maximum of two shifts a day, each of which is a net of eight working hours. Yet, there are a few thousand late containers every year that costs the company a great deal.

While tardiness is costly, labor cost is also another concern. Intelligent selection of incoming dock doors can significantly reduce labor costs. The company tends to over or under estimate the amount of dock doors to select, which in turn, increases their labor costs. The company ends up either paying for unneeded labor by overestimating labor need, or paying tardiness cost from underestimating labor need.

Despite their current efforts on managing their operations, it is still challenging to handle the large amount of daily containers over limited resources, let alone trading off between the costs resulting from interdependent decisions such as scheduling and selection. Their annual cost resulting from both scheduling and selection is in the six figures. Therefore, providing a scheduling and selection decision tool is essential to saving a portion of the annual operational cost, not to mention, the insights this tool can provide on a daily basis supporting other operational decisions.

Scheduling vehicles (e.g., trucks, containers or trailers) has been studied in the context of cross-docking in terms of scheduling over single inbound/outbound doors, multiple dock doors and inbound doors only. Wang and Regan [59] study the problem of scheduling incoming trailers over inbound dock doors to reduce the time goods spend inside the cross-dock. They introduce dynamic simulation models and solve the problem using two time-based algorithms. Rosales et al. [48] study the problem of assigning incoming trucks to inbound dock doors and selecting dock doors for processing incoming trucks with the objective of minimizing the operational cost. Their operational cost consists of traveling distance and labor costs. Since one worker is allowed per dock door, minimizing the number of dock doors helps with minimizing the labor cost. They develop a mixed-integer programming (MIP) model that minimizes the operational cost and provides a balanced workload to all workers in one shift for a large cross-dock in Georgetown. Chmielewski et al. [19] present a MIP model that minimizes travel distance and solve the problem using decomposition and column-generation approaches. The authors also obtain Pareto-optimal solutions that reduce waiting time optimally. Liao et al. [35] study the problem of scheduling incoming trucks over inbound dock doors. They present a model that focuses on the minimization of the total weighted tardiness by simultaneously assigning and sequencing incoming trucks. The authors use simulated annealing, Tabu search, ant colony optimization, decomposition evolution, and two hybrid decomposition evolution algorithms to solve the problem. Boysen et al. [12] consider an incoming truck scheduling problem that arises in the postal industry with predetermined departures of outgoing trucks to minimize the total lost profit resulting from unfulfilled/late shipments. They introduce a MIP model along with decomposition procedures and simulated annealing heuristics. However, none of this work has considered the selection of inbound dock doors simultaneously with the scheduling (i.e., simultaneous assignment and sequencing) decisions. Van Belle et al. [58] provide a comprehensive review on operational problems arise in cross-docking including scheduling problems, while Boysen [10] provide a detailed literature review on cross-dock scheduling problems. Other reviews on cross-docking are provided by Agustina et al. [2], Shuib and Fatthi [53], and Buijs et al. [14]. Recently, Ladier and Alpan [32] proposed a framework that highlights the gaps between the literature and some cross-docking practices in France.

The container scheduling and dock door selection problems arise in other applications apart from cross-docking. First, the fundamental container scheduling problem is equivalent to a classical parallel machine scheduling problem, where jobs get assigned and sequenced over parallel machines to minimize some performance criterion such as tardiness, earliness, lateness or completion time (Pinedo [47]). Second, the integration of dock door selection with container scheduling has been studied in the context of machine availability constraints with the assumption that machines are subject to maintenance periods or unexpected breakdowns. The former is modeled deterministically and largely with the assumption that the predetermined maintenance periods are known beforehand, without being part of the decision process as seen in Ma et al. [38], Lee et al. [33], and Sanlaville and Schmidt [51]. The latter is modeled stochastically as seen in Adiri et al. [1] and Liu and Sanlaville [36]. To the best of our knowledge, the only work that takes into account scheduling and selection simultaneously, in a machine environment, is carried out by Cao et al. [15], Fanjul-Peyro and Ruiz [24] and Alidaee and Li [4]. Among these studies, only Cao et al. [15] consider the selection of machines and schedule of jobs with the objective of minimizing the total weighted tardiness and machine setup/usage cost. The authors introduce the problem of scheduling jobs and selecting machines to process these jobs in order to minimize the total weighted tardiness cost of late jobs and the setup cost of using the machines. They present a network based model that solves small size instances optimally, and a Tabu search heuristic that solves instances up to 60 jobs and 6 machines. However, they only consider a static problem of scheduling and selecting in a single period time horizon.

The main contribution of this paper is in introducing two new integer programming (IP) models that accommodate for static and dynamic environments of the *container scheduling and dock door selection problem* (CSDS). First, although the static CSDS has been modeled before by Cao et al. [15] in terms of machine scheduling, our newly introduced CSDS model outperforms the existing one in terms of CPU time, branch and bound nodes, and solving larger set of instances optimally. Second, we introduce a dynamic CSDS that takes into account scheduling containers and selecting doors over multiple periods of time that involve shifts or days. Such a dynamic modeling reflects real life applications of cross-docking as it provides resource planning that spans over multiple periods instead of just one. We introduce new sets of instances, based on our observations on a large cross-dock, with up to 100 incoming containers and 40 inbound dock doors at a given day.

Our static CSDS significantly outperforms the existing one, and our dynamic CSDS is able to solve almost all the instances optimally in a reasonable time.

The rest of this paper is organized as follows: In Section 4.2, we formally define the basic container scheduling problem and its associated IP model. In Section 4.3, we introduce the static CSDS, present the existing MIP model by Cao et al. [15], and introduce a new IP model for the same problem. In Section 4.4, we introduce the dynamic CSDS along with a new IP model that takes into account scheduling containers and selecting dock doors over several shifts along the planning time horizon. Finally, computational experiments are reported in Section 4.5, and conclusions with future research directions are provided in Section 4.6.

4.2 The Container Scheduling Problem

In order to lay out our integrated container scheduling models, we first define formally the basic container scheduling problem with all its mathematical notations and an existing model. Then, in the following sections, we introduce the integration of selection decisions and notation.

Let N and K denote the sets of incoming containers and inbound doors, respectively. Let p_{ik} denote the processing time to unload container $i \in N$ at door $k \in K$, and d_i denote the due date of returning empties i to the port. Each container i has a weight reflecting its importance or priority and is denoted by w_i. All containers are available at the yard for scheduling at the beginning of the day, and hence, the ready time is always zero. The completion time is a continuous variable denoted as C_i, while tardiness T_i is an integer variable that indicates by how many hours a container is late. The container scheduling problem is defined as the assignment and sequencing of each incoming container over exactly one inbound dock door at a given time such that containers do not overlap over the same dock door and the same time and the total weighted tardiness is minimized. The weighted tardiness of each container is computed as: $w_i \mathrm{T}_i = w_i \max\{C_i - d_i, 0\}$, where w_i represents the weight of container i.

Unlu and Mason [57] evaluate empirically four existing MIP models for parallel machine scheduling problems. They demonstrate that a time index model is the best in terms of providing tight LP bounds and optimal solutions for larger instances as compared to all the other three models. The time indexed model relies on the discretization of the time horizon into time periods with an upper bound on the last job's completion time. Denote by T the set of discretized time horizon. For every $i \in N$, $k \in K$, and $t \in T$, we define the following decision variables

$$x_{ikt} = \begin{cases} 1, & \text{if container } i \text{ starts processing on inbound door } k \text{ at time } t \\ 0, & \text{otherwise.} \end{cases}$$

Instead of explicitly calculating the completion time of a container and its tardiness, the time indexed model allows to pre-process such information, reducing the number of constraints and variables needed as well as strengthening the LP bounds. For every $i \in N$, $k \in K$, and $t \in T$, the total weighted tardiness of a container can be stated as $\alpha_{ikt} = w_i \max\{0, t + p_{ik} - d_i\}$. The container scheduling problem can be stated as:

$$[M_0] \qquad \text{minimize} \quad \sum_{i \in N} \sum_{k \in K} \sum_{t \in T} \alpha_{ikt} x_{ikt} \tag{4.1}$$

$$\text{subject to:} \quad \sum_{k \in K} \sum_{t \in T} x_{ikt} = 1 \qquad i \in N \tag{4.2}$$

$$\sum_{i \in N} \sum_{h=\max\{0, t-p_{ik}\}}^{t-1} x_{ikh} \leq 1 \qquad k \in K, t \in T \tag{4.3}$$

$$x_{ikt} \in \{0,1\} \qquad i \in N, k \in K, t \in T. \tag{4.4}$$

The objective function (4.1) seeks to minimize the total weighted tardiness. Constraints (4.2) make sure that each container is assigned exactly to one inbound dock door at a unique time. Constraints (4.3) make sure that each door at a given time can handle at most one container. Finally, constraints (4.4) impose binary conditions on the time indexed variables. We use this model as the foundation to integrate the container scheduling with the dock door selection decisions in Sections 4.3 and 4.4.

4.3 The Static Container Scheduling and Dock Door Selection Problem

In order to integrate container scheduling with dock door selection decisions, we introduce the following notation. Let β_k denote the fixed labor cost associated with dock door k. The static CSDS problem is defined as the assignment and sequencing of each container i to exactly one inbound dock door k at a given time t such that the containers do not overlap over the same dock door and time, and dock door k is selected for processing containers, with the objective of minimizing the total operational cost. The total operational cost consists of total weighted tardiness and labor costs, resulting from the simultaneous scheduling and selection decisions. For every dock door $k \in K$, we introduce the following selection decision variable

$$z_k = \begin{cases} 1, & \text{if inbound dock door } k \text{ is selected} \\ 0, & \text{otherwise.} \end{cases}$$

The static CSDS can be stated as

$$[M_1] \quad \text{minimize} \quad \sum_{i \in N} \sum_{k \in K} \sum_{t \in T} \alpha_{ikt} x_{ikt} + \sum_{k \in K} \beta_k z_k \tag{4.5}$$

$$\text{subject to:} \quad \sum_{k \in K} \sum_{t \in T} x_{ikt} = 1 \qquad i \in N \tag{4.2}$$

$$\sum_{i \in N} \sum_{h=\max\{0,t-p_{ik}\}}^{t-1} x_{ikh} \le z_k \qquad k \in K, t \in T \tag{4.6}$$

$$x_{ikt} \in \{0,1\} \qquad i \in N, k \in K, t \in T \tag{4.4}$$

$$z_k \in \{0,1\} \qquad k \in K. \tag{4.7}$$

The objective function (4.5) seeks to minimize the total operational cost: tardiness and labor costs. The right hand side of resource constraints (4.6) is modified by the new selection decision variables as opposed to constraints (4.3). Finally, constraints (4.7) impose integrality conditions on the dock door selection variables.

Alternatively, Cao et al. [15] introduce a network-based model, and define the following variables in the context of machine scheduling. Without loss of generality, we modify their notations and model in accordance with a cross-docking context. Let $x'_{ijk} = 1$ if container j immediately follows container i on door k, 0 otherwise. Let $y_{jk} = 1$ if container j is assigned to door k, 0 otherwise. Using the network-based model presented in Cao et al. [15], the static CSDS model can thus be formulated as

$$[N_1] \quad \text{minimize} \quad \sum_{i \in N} w_i \max\{0, C_i - d_i\} + \sum_{k \in K} \beta_k z_k \tag{4.8}$$

$$\text{subject to} \quad \sum_{i \in N: i \neq j} \sum_{k \in K} x'_{ijk} = 1 \qquad j \in N \tag{4.9}$$

$$\sum_{i \in N: i \neq j} x'_{ijk} = y_{jk} \qquad j \in N, k \in K \tag{4.10}$$

$$\sum_{j \in N: j \neq i} x'_{ijk} \leq y_{ik} \qquad i \in N, k \in K \tag{4.11}$$

$$\sum_{j \in N} x'_{0jk} \leq z_k \qquad k \in K \tag{4.12}$$

$$C_j + M\left(1 - x'_{ijk}\right) \geq C_i + p_{jk} \qquad j \in N, i \in N, k \in K \tag{4.13}$$

$$x'_{ijk} \in \{0,1\} \qquad (i,j) \in N, k \in K \tag{4.14}$$

$$y_{jk} \in \{0,1\} \qquad j \in N, k \in K \tag{4.15}$$

$$z_k \in \{0,1\} \qquad k \in K \tag{4.7}$$

$$C_i \geq 0 \qquad i \in N. \tag{4.16}$$

The objective (4.8) seeks to minimize the total weighted tardiness cost and the cost of selecting doors. Constraints (4.9) ensure that each container must be processed at one machine in one position. Constraints (4.10) state that container j should immediately follow another container on door k if it is placed on this door. Constraints (4.11) state that if container i is processed on door k, it will be immediately followed by at most one another container assigned to the same door. Constraints (4.12) make sure that only one container can follow the dummy container on any selected door k. The completion time is calculated via constraints (4.13), where big M is introduced as a very large positive number. Constraints (4.14), (4.15),(4.7) impose binary restrictions on the variables, while constraints (4.16) define the positive continuous range of the completion time variables.

The models M_1 and N_1 differ from M_0 by adding decision variables of selecting dock doors to be open or closed in the whole planning time horizon. This has an impact on trading off between tardiness and labor cost. In M_0, we assume that all dock doors are selected and available for unloading containers, which is not the case in practice as companies tend to close some dock doors when the flow can be managed with less labor cost. M_1 is thus a generalization of M_0.

Finally, one important difference between the time index model, M_1, and network-based model, N_1, is that the former assumes integer values when it comes to processing times, while the latter can take integer and real numbers for its processing times. In Section 4.5, we compare the performances of both static CSDS models, M_1 and N_1, using the set of instances introduced by Cao et al. [15], by slightly modifying their processing times to be integer. We also introduce a new larger set of instances to test both models.

4.4 The Dynamic Container Scheduling and Dock Door Selection Problem

We next introduce a time indexed model for the dynamic CSDS. The dynamic model is a generalization of the static one (i.e., M_1) introduced in the previous section. In a dynamic time horizon, containers get scheduled over several shifts, and dock doors get selected over several shifts too. Let S denote the set of shifts in which the time horizon T is partitioned, and let $T(s) \subseteq T$ denote the set of times $t \in T$ that belong to shift s. An interesting and challenging feature of the dynamic CSDS is that, depending on the starting processing time of a container, its processing may span more than one shift and thus, the model needs to ensure that the required shifts are selected. We modify the variables z_k to become z_{ks} to indicate whether door k is selected or not in shift s, and similarly, β_k becomes β_{ks} for the labor cost. The dynamic CSDS problem can thus be stated as

$$[M_2] \qquad \text{minimize} \quad \sum_{i\in N}\sum_{k\in K}\sum_{t\in T}\alpha_{ikt}x_{ikt} + \sum_{k\in K}\sum_{s\in S}\beta_{ks}z_{ks} \tag{4.17}$$

$$\text{subject to:} \quad \sum_{k\in K}\sum_{t\in T} x_{ikt} = 1 \qquad i\in N \tag{4.2}$$

$$\sum_{i\in N}\sum_{h=\max\{0,t-p_{ik}\}}^{t-1} x_{ikh} \le z_{ks} \qquad k\in K, s\in S, t\in T(s) \tag{4.18}$$

$$\sum_{t\in\bigcup_{r=1}^{s}T(r):t+p_{ik}-1\in\bigcup_{r=s}^{|S|}T(r)} x_{ikt} \le z_{ks} \qquad i\in N, k\in K, s\in S \tag{4.19}$$

$$x_{ikt}\in\{0,1\} \qquad i\in N, k\in K, t\in T \tag{4.4}$$

$$z_{ks}\in\{0,1\} \qquad k\in K, s\in S. \tag{4.20}$$

The objective function (4.17) is a generalization of (4.5) to include the selection decisions for each shift in the time horizon and the labor cost associated with selecting a dock door in a given shift. Constraints (4.18) generalize constraints (4.6) to include the shifts, and the time periods associated with each shift, $T(s)$. We introduce constraints (4.19) to make sure that a container can only be assigned to a door if selected at a given shift, and if that same door stays open throughout a number of shifts until a container is completely processed. Finally, binary conditions are imposed on the generalized selection variables in (4.20).

4.5 Computational Experiments

We next present the results of computational experiments performed to assess the behavior of the static and dynamic CSDSs introduced in this paper. All models are implemented using the callable library of CPLEX 12.7.0 with four threads. We first explain the data generation mechanism of three different sets of instances. We then computationally compare the static CSDS models introduced in this paper, M_1, and the one introduced by Cao et al. [15], N_1. Detailed computational experiments are then reported on the dynamic CSDS, M_2, with scattered and common due dates, respectively. Finally, a cost parametric analysis is provided to demonstrate how the objective function trades off between both tardiness and labor costs based on the priority given to each.

4.5.1 Data Generation

The first data set (i.e., SetA) consists of six examples provided by Cao et al. [15]. The authors generate instances with 20 containers scheduled over three doors with various processing times, due dates, variant weights and labor costs. Whenever a processing time value is fractional, we round it to the closest integer value in order to test our static time index model as well. The second data set (i.e., SetB) is introduced in this paper and generated as follows:

- processing time: based on the data provided by the company, processing times of containers follow a log-normal distribution $p_{ik} \sim \lceil \text{LOGNORMAL}(\mu, \sigma)/\ell_k \rfloor$ with $\mu = 1.25$ and $\sigma = 0.76$, bounded by 24 as the maximum number of hours a container can take to be unloaded. The parameter $\ell_k \sim \text{UNIFORM}(1, 3)$ indicates the laborer skill level for each dock door. These skill levels are needed to calculate each container's processing time, based upon the dock door to which it is assigned. Furthermore, we round all processing times to the closest integer value.

- due date: we adopt the data generation mechanism in Chen and Powell [18] and Ho and Chang [30]. The due date of a container is stated as $d_i = \max\{a_i, b_q\}$, where $a_i = \min\limits_{k \in K} p_{ik}$, and b_q is an integer uniformly distributed in the interval $[1, 6r/q]$ with $r = |N|/|K|$ and q

being a controllable parameter. The larger the q the tighter the due dates are. We consider $q = 1, 2, 3, 4, 5$ for each group of instances.

- weight: the weight or importance of each container can be classified as rush, hot or normal. Hence, it is generated as $w_i \sim$ UNIFORM $(1, 2, 3)$.

- labor cost: we assume that the cost $\beta_{ks} = \beta_k$ for all shifts, and only differ by doors. That is $\beta_k = [b + \gamma_k]\, s$, where b is the basic labor cost added to the skill level $\gamma_k \sim$ UNIFORM $\{1, 2, 3\}$. Since each shift consists of 8 hours, i.e., $s = 8$, we multiply the term by 8 to estimate the labor cost per shift. All time units are measured in hours and the tardiness is counted in days, where each day consists of one shift. Since each day consists of one shift of 8 hours, the objective term associated with tardiness is multiplied by $1/8$. This way the cost of tardy container per day, \$50, is only considered when a container is late by number of days/shifts.

In the following, we consider groups of 9 sets of instances, each of which considers different due dates based on the parameter q with a total of 45 instances per set. The third set of instances (i.e. SetC) is generated the same way except for two common due dates assumed for each instance. In all subsequent tables, we use the following performance measurements:

- %LP: refers to the LP deviation from the optimal value. That is $\%LP = \frac{|OPT - LP|}{OPT} 100$.

- %gap: refers to the optimality gap upon termination. That is $\%gap = \frac{|OPT - LB|}{OPT} 100$. Where LB is the best lower bound obtained.

- %tardy: it is the percentage of the tardiness cost from the total operational cost. This cost is a result of the optimal/best schedule found.

- %labor: it is the percentage of the labor cost from the total operational cost. This cost is a result of the optimal/best selection found.

- %nodes: it is the number of branch-and-bound nodes CPLEX takes to reach at the optimal/best solution found.

- time: it is measured in seconds. The time limit varies from one table to another. We will refer to those limits as we proceed.

- selected: indicates the number of dock door selected to unload containers in a given optimal/incumbent solution.

Other specific performance measurements are reported whenever they show up as they differ from one experiment to another.

4.5.2 A Comparison Between the Static CSDSs

Tables 4.1, 4.2 and 4.3 compare the performance of both static CSDSs, N_1 introduced by Cao et al. [15], and M_1 introduced in this paper using SetA and SetB, respectively. In addition to the aforementioned performance measurements, we add $\%UB = \frac{|OPT-UB|}{OPT}100$ to indicate the percentage deviation of the upper bounds found with respect to the optimal/incumbent value.

Table 4.1: Static CSDSs - SetA from Cao et al. [15]: N_1 vs. M_1

	N_1								M_1							
Examples	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected
6x3E1	100	0.00	0.00	14.29	85.71	4,420	0.98	2/3	2.22	0.00	0.00	14.29	85.71	0	0.44	2/3
6x3E2	100	0.00	0.00	14.29	85.71	4,489	0.91	2/3	2.04	0.00	0.00	14.29	85.71	0	0.44	2/3
6x3E3	100	0.00	0.00	27.45	72.55	3,753	0.82	1/3	6.98	0.00	0.00	27.45	72.55	0	0.77	1/3
6x3E4	100	100	22.73	0.00	100	127,703,926	day	2/3	5.40	0.00	0.00	0.00	100	62	9.23	2/3
6x3E5	100	100	22.73	0.00	100	126,984,070	day	2/3	3.75	0.00	0.00	0.00	100	0	6.35	2/3
6x3E6	100	100	0.00	0.00	100	140,496,755	day	3/3	22.11	0.00	0.00	0.00	100	0	5.89	3/3
average	100	50.00	7.58	9.34	90.66	65,866,236	0.90	2/5	7.08	0.00	0.00	9.34	90.66	10	3.85	2/5

Table 4.1 shows clearly that model M_1 outperforms N_1 of Cao et al. [15] among all performance measurements, even after allowing a time limit of a day. This is mostly due to the strong LP bounds provided by the time indexed models. Moreover, the last three instances indicate a tardiness cost of zero, indicating ample due dates. We mitigate such behavior in the next table by introducing the data SetB that ranges from tight to ample due dates, reflecting a better balance between tardiness and labor costs.

Tables 4.2 and 4.3 compare again our static model M_1 to the one introduced by Cao et al. [15], i.e., N_1 in a larger and varied set of instances. The %LP shows consistently a cost of zero in all the instances when it comes to N_1, whereas an average %LP of %1.43 for M_1. We limit the run time here to 2 hours after several experiments that show no significant improvement, when run on

Table 4.2: Static CSDSs - SetB: N_1 vs. M_1

		N_1								M_1							
instance	q	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected
20x5	1	100	100	2.01	0.00	100	13,148,001	2 hours	2/5	20.44	0.00	0.00	3.36	96.64	102	2.76	2/5
	2	100	89.55	3.48	48.82	51.18	8,795,695	2 hours	2/5	2.06	0.00	0.00	47.04	52.96	2,652	3.98	2/5
	3	100	99.98	7.67	27.51	72.49	6,031,201	2 hours	3/5	1.74	0.00	0.00	47.04	52.96	0	0.80	2/5
	4	100	94.68	0.00	46.10	53.90	7,032,164	2 hours	2/5	1.10	0.00	0.00	46.10	53.90	0	0.48	2/5
	5	100	92.58	1.48	55.56	44.44	7,690,996	2 hours	2/5	0.39	0.00	0.00	54.90	45.10	0	0.53	2/5
30x5	1	100	100	69.41	0.00	100	6,281,298	2 hours	4/5	2.90	0.00	0.00	5.88	94.12	0	2.33	2/5
	2	100	100	18.87	15.32	84.68	3,738,900	2 hours	4/5	1.75	0.00	0.00	23.18	76.82	195	3.21	3/5
	3	100	100	12.83	31.32	68.68	3,837,000	2 hours	4/5	0.74	0.00	0.00	39.27	60.73	0	2.64	3/5
	4	100	100	14.18	33.77	66.23	4,051,907	2 hours	4/5	1.66	0.00	0.00	42.29	57.71	1,868	8.74	3/5
	5	100	100	16.35	39.71	60.29	4,268,201	2 hours	4/5	1.00	0.00	0.00	45.02	54.98	32	2.88	3/5
40x5	1	100	100	41.14	6.07	93.93	5,105,400	2 hours	3/5	0.98	0.00	0.00	8.57	91.43	64	5.50	2/5
	2	100	100	10.77	11.11	88.89	5,535,764	2 hours	4/5	3.31	0.00	0.00	26.15	73.85	71,608	210.32	3/5
	3	100	100	8.31	25.58	74.42	4,037,739	2 hours	4/5	1.14	0.00	0.00	21.41	78.59	1,052	11.11	4/5
	4	100	100	10.07	20.33	79.67	4,455,100	2 hours	5/5	1.28	0.00	0.00	30.20	69.80	11,570	41.82	4/5
	5	100	100	15.40	44.48	55.52	3,749,000	2 hours	4/5	0.75	0.00	0.00	35.93	64.07	3,559	15.94	4/5
50x10	1	100	100	42.46	3.24	96.76	1,075,684	2 hours	6/10	6.86	0.00	0.00	1.54	98.46	120,709	2335.78	4/10
	2	100	100	20.11	29.95	70.05	1,283,279	2 hours	6/10	0.80	0.00	0.00	27.68	72.32	33,321	511.38	5/10
	3	100	100	22.32	26.46	73.54	621,647	2 hours	7/10	1.36	0.00	0.00	33.22	66.78	133,595	2194.90	5/10
	4	100	100	24.61	35.00	65.00	610,407	2 hours	7/10	0.47	0.00	0.00	38.94	61.06	48	21.94	5/10
	5	100	100	19.17	36.61	63.39	456,503	2 hours	7/10	0.43	0.00	0.00	45.06	54.94	164	32.16	5/10
60x15	1	100	100	45.08	1.58	98.42	337,374	2 hours	8/15	1.41	0.76	0.00	10.30	89.70	711,944	2 hours	5/15
	2	100	100	19.79	21.90	78.10	378,042	2 hours	8/15	0.75	0.00	0.00	29.24	70.76	1,682	134.92	6/15
	3	100	100	17.40	27.94	72.06	277,587	2 hours	8/15	0.69	0.00	0.00	36.81	63.19	14,270	635.86	6/15
	4	100	100	17.73	27.20	72.80	356,597	2 hours	9/15	0.65	0.31	0.00	41.87	58.13	566,098	2 hours	6/15
	5	100	100	18.02	42.96	57.04	249,551	2 hours	8/15	1.01	0.27	0.00	49.08	50.92	602,180	2 hours	6/15

Table 4.3: Static CSDSs - SetB: N_1 vs. M_1

		N_1								M_1							
instance	q	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected	%LP	%gap	%UB	%tardy	%labor	nodes	time	selected
60x20	1	100	100	39.83	9.09	90.91	285,649	2 hours	8/20	0.53	0.00	0.00	16.95	83.05	13	200.40	5/20
	2	100	100	28.01	28.28	71.72	268,901	2 hours	9/20	0.64	0.00	0.00	35.52	64.48	6,572	559.94	6/20
	3	100	100	29.17	34.93	65.07	168,005	2 hours	9/20	0.64	0.00	0.00	42.65	57.35	14,749	1120.74	6/20
	4	100	100	22.55	45.77	54.23	159,077	2 hours	8/20	0.34	0.00	0.00	49.08	50.92	920	189.38	6/20
	5	100	100	17.41	47.27	52.73	117,628	2 hours	8/20	0.41	0.00	0.00	51.93	48.07	57,240	3021.23	6/20
70x20	1	100	100	43.93	18.07	81.93	69,044	2 hours	11/20	0.31	0.00	0.00	20.23	79.77	404	282.63	7/20
	2	100	100	31.16	32.95	67.05	115,784	2 hours	11/20	0.64	0.00	0.00	41.09	58.91	7,195	830.50	7/20
	3	100	100	29.12	38.69	61.31	70,116	2 hours	11/20	0.84	0.00	0.00	41.39	58.61	12,212	1275.27	8/20
	4	100	100	24.77	45.09	54.91	55,402	2 hours	11/20	0.54	0.00	0.00	48.28	51.72	289,287	2 hours	8/20
	5	100	100	23.62	51.81	48.19	61,613	2 hours	10/20	0.45	0.20	0.00	50.04	49.96	369,693	2 hours	8/20
80x30	1	100	100	1065.06	98.38	1.62	11,431	2 hours	2/30	0.49	0.00	0.00	31.60	68.40	632	708.14	7/30
	2	100	100	804.70	98.45	1.55	11,970	2 hours	2/30	0.51	0.15	0.00	42.44	57.56	191,064	2 hours	8/30
	3	100	100	368.56	94.87	5.13	15,757	2 hours	4/30	0.31	0.00	0.00	47.96	52.04	143,721	2 hours	8/30
	4	100	100	723.85	98.56	1.44	9,715	2 hours	2/30	0.02	0.00	0.00	51.21	48.79	0	460.30	8/30
	5	100	100	468.02	98.23	1.77	13,208	2 hours	2/30	0.36	0.00	0.00	53.91	46.09	2,682	1290.02	8/30
100x40	1	100	100	1174.33	95.09	4.91	1,306	2 hours	10/40	0.21	0.20	0.00	31.08	68.92	65,272	2 hours	10/40
	2	100	100	1110.68	98.33	1.67	1,520	2 hours	4/40	0.25	0.16	0.00	44.01	55.99	50,641	2 hours	10/40
	3	100	100	976.65	98.32	1.68	720	2 hours	4/40	0.47	0.37	0.00	49.81	50.19	58,220	2 hours	10/40
	4	100	NA	NA	NA	NA	3,714	2 hours	0/40	0.39	0.12	0.00	48.94	51.06	67,760	2 hours	11/40
	5	100	100	854.67	98.34	1.66	1,654	2 hours	4/40	0.13	0.00	0.00	52.41	47.59	2,891	4720.72	11/40
average		100	99.47	188.97	42.93	57.07	2,107,717	2 hours	6	1.43	0.06	0.00	36.46	63.54	80,397	612.92	6

a 24 hour period. The optimality gap %gap is close to zero on average for M_1, while rarely any improvement found in the gap, when it comes to N_1. The previous two tables are strong indicators of the good performance of the time index model M_1, when compared with N_1.

4.5.3 The Dynamic CSDS

We next use the data and SetC to investigate the performance of the dynamic CSDS using scattered due dates and common due dates, respectively. Unlike the static CSDS, here the CPU time limit is set to one day, since our experiments show the benefit of adding more time than just two hours. The CPU times are reported in seconds, unless they are as long as a day. In addition to the performance measurements we introduced in Section 4.5.1, we introduce three new measurements, because we are now evaluating a dynamic model, where more statistical data can provide insights on the solutions. We report on the maximum completion time among all containers in hours, C_{max}, as well as the associated last shift used where the last container was unloaded, S_{max}. We also report on the percentage of shifts used to unload containers, $\%shifts$.

Tables 4.4 and 4.5 provide the results associated with SetB. We note that 42 out 45 instances of the dynamic CSDS problem using M_2 were solved to optimality. The %LP gets tighter as the size of instances increases, while the optimality %gap is 0.16 on average. It is clear that the %tardy increases as the q parameter increases, indicating tighter due dates on containers. It is interesting to mention that only in one instance $40x5 : q = 5$ where the third shift is not fully utilized. In other words, the last container was emptied one hour before the end of the last shift. However, our model is not intended to optimize such metric. In Section 4.5.5, we demonstrate how the cost functions vary depending on the priority given to the tardiness and labor costs, respectively.

We next report on a data SetC with common due dates in Table 4.6. We observe that 33 out 45 instances were solved optimally for the dynamic CSDS problem using M_2. We notice that generally the looser the due dates, the more difficult to obtain optimal solutions in the given time limit. This behavior could be because of the quality of feasible solutions obtained during the branch-and-bound tree. When due dates are loose, there is more symmetry and many possible feasible solutions that are considered, are not useful. Of course, this could be improved by some add-hock heuristics embedded during the branching process. However, this is not within the scope of this paper.

Table 4.4: Dynamic CSDS - SetB Scattered Due Dates: M_2

instance	q	%LP	%gap	%tardy	%labor	nodes	time	C_{max}	S_{max}	%shifts	selected
20x5	1	6.05	0.00	0.00	100	12,899	27.68	24	3	34	2/5
	2	7.58	0.00	12.20	87.80	4,139	13.45	16	2	60	5/5
	3	7.72	0.00	21.47	78.53	1,714	8.90	24	3	34	3/5
	4	7.72	0.00	13.08	86.92	1,156	7.39	16	2	60	5/5
	5	7.39	0.00	30.50	69.50	4,950	16.04	16	2	9	4/5
30x5	1	5.17	0.00	0.00	100	38,471	261.06	32	4	45	4/5
	2	1.52	0.00	6.07	93.93	386	7.72	24	3	60	4/5
	3	1.39	0.00	14.71	85.29	3,614	23.66	24	3	60	4/5
	4	1.27	0.00	15.23	84.77	96	5.28	24	3	60	4/5
	5	0.64	0.00	17.73	82.27	213	5.63	24	3	60	4/5
40x5	1	2.39	0.00	12.53	87.47	5,267,372	44053.26	40	5	14	5/5
	2	4.57	0.00	15.28	84.72	47,571	448.50	23	3	13	5/5
	3	2.60	0.00	16.51	83.49	4,268	64.30	16	2	14	5/5
	4	3.24	0.00	21.13	78.87	152	9.74	16	2	14	5/5
	5	2.64	0.00	23.06	76.94	5,846	46.21	16	2	14	5/5
50x10	1	1.28	0.00	0.00	100	84,404	2363.73	24	3	40	7/10
	2	0.46	0.00	10.94	89.06	3,757	152.80	24	3	40	7/10
	3	0.69	0.00	12.61	87.39	486	44.98	24	3	40	7/10
	4	0.71	0.00	16.87	83.13	1,420	68.50	24	3	40	7/10
	5	0.24	0.00	20.75	79.25	0	18.16	24	3	40	7/10
60x15	1	3.81	3.51	0.00	100	2,508,127	day	24	3	38	10/15
	2	2.60	1.62	5.75	94.25	4,353,896	day	16	2	57	10/15
	3	2.54	1.98	9.95	90.05	3,100,429	day	16	2	57	10/15
	4	1.87	0.00	13.16	86.84	2,001,383	day	16	2	57	10/15
	5	1.88	0.00	19.11	80.89	301,387	28263.97	16	2	57	10/15

Table 4.5: Dynamic CSDS - SetB Scattered Due Dates: M_2

instance	q	%LP	%gap	%tardy	%labor	nodes	time	C_{max}	S_{max}	%shifts	selected
	1	1.55	0.00	0.00	100	401,887	36371.11	16	2	38	9/20
	2	1.01	0.00	9.43	90.57	41,812	3013.42	16	2	38	11/20
60x20	3	0.93	0.00	14.79	85.21	1,127,246	71988.10	16	2	38	11/20
	4	1.48	0.00	16.29	83.71	65,637	5901.06	16	2	40	13/20
	5	1.13	0.00	19.63	80.37	17,297	1815.20	16	2	40	13/20
	1	0.97	0.00	3.42	96.58	129,149	17416.27	24	3	34	10/20
	2	0.41	0.00	13.63	86.37	75,303	10555.84	24	3	34	10/20
70x20	3	0.50	0.00	18.62	81.38	20,118	3262.76	24	3	34	10/20
	4	0.80	0.00	24.18	75.82	138,758	15752.81	24	3	34	10/20
	5	0.58	0.00	26.34	73.66	175,510	27117.38	24	3	34	10/20
	1	0.56	0.00	7.11	92.89	161,710	45097.69	24	3	24	12/30
	2	0.32	0.00	14.23	85.77	57,167	11319.32	24	3	25	15/30
80x30	3	0.55	0.00	17.38	82.62	389,889	84619.61	24	3	25	15/30
	4	0.44	0.00	21.05	78.95	85,223	20519.91	24	3	25	15/30
	5	0.20	0.00	23.20	76.80	15,609	4601.72	24	3	25	15/30
	1	0.13	0.00	3.03	96.97	707	1870.29	16	2	34	22/40
	2	0.24	0.00	9.96	90.04	240,314	day	16	2	34	22/40
100x40	3	0.47	0.00	12.13	87.87	290,544	day	16	2	35	25/40
	4	0.43	0.00	16.27	83.73	74,497	46813.68	16	2	35	25/40
	5	0.52	0.00	19.88	80.12	31,465	26408.68	16	2	35	25/40
average		2.03	0.16	13.76	86.24	473,066	22861.24	21	3	37	10

Table 4.6: Dynamic CSDS - SetC Common Due Dates: M_2

instance	q	%LP	%gap	%tardy	%labor	nodes	time	C_{max}	S_{max}	%shifts	selected
20x5	1	0.42	0.00	0.00	100	0	1.53	16	2	60	4/5
	2	0.00	0.00	4.07	95.93	0	0.94	16	2	60	4/5
	3	0.00	0.00	8.70	91.30	0	0.41	16	2	60	4/5
	4	0.00	0.00	12.92	87.08	0	0.97	16	2	60	4/5
	5	0.00	0.00	14.49	85.51	0	1.16	16	2	60	4/5
30x5	1	3.13	0.00	0.00	100	338	7.04	32	4	40	2/5
	2	2.78	0.00	11.11	88.89	2,057	10.37	32	4	40	2/5
	3	3.15	0.00	12.13	87.87	92,225	385.80	32	4	45	4/5
	4	3.32	0.00	16.91	83.09	549,916	2111.69	32	4	45	4/5
	5	1.70	0.00	21.64	78.36	893	10.17	31	4	47	4/5
40x5	1	2.02	0.00	5.55	94.45	162,711	1519.45	72	9	27	2/5
	2	1.59	0.00	11.49	88.51	99,789	952.13	56	7	40	3/5
	3	1.56	0.00	20.00	80.00	220,489	1814.24	56	7	40	4/5
	4	1.30	0.00	24.09	75.91	18,598	268.94	56	7	40	4/5
	5	1.18	0.00	23.68	76.32	48,674	392.03	48	6	50	5/5
50x10	1	0.39	0.00	0.00	100	19	34.71	24	3	44	6/10
	2	1.34	0.00	2.40	97.60	52,324	1493.78	24	3	44	6/10
	3	3.04	0.23	4.06	95.94	12,516,921	day	16	2	70	10/10
	4	2.60	0.15	5.34	94.66	4,788,088	day	16	2	70	10/10
	5	2.60	0.00	5.34	94.66	36,077	990.38	16	2	70	10/10
60x15	1	0.00	0.00	0.00	100	0	29.63	24	3	34	7/15
	2	0.00	0.00	6.03	93.97	0	35.01	24	3	34	7/15
	3	0.00	0.00	13.67	86.33	0	29.28	24	3	34	7/15
	4	0.03	0.00	20.44	79.56	0	47.87	24	3	34	7/15
	5	0.00	0.00	25.75	74.25	0	30.51	24	3	34	7/15

Table 4.7: Dynamic CSDS - SetC Common Due Dates: M_2

instance	q	%LP	%gap	%tardy	%labor	nodes	time	C_{max}	S_{max}	%shifts	selected
	2	0.00	0.00	6.03	93.97	0	35.01	24	3	34	7/15
	3	0.00	0.00	13.67	86.33	0	29.28	24	3	34	7/15
	4	0.03	0.00	20.44	79.56	0	47.87	24	3	34	7/15
	5	0.00	0.00	25.75	74.25	0	30.51	24	3	34	7/15
	1	2.24	2.07	0.00	100	1,390,449	day	16	2	38	12/20
	2	1.03	0.32	2.52	97.48	2,264,198	day	16	2	38	12/20
60x20	3	0.66	0.26	8.66	91.34	4,181,407	day	16	2	38	12/20
	4	0.51	0.00	9.02	90.98	2,529,273	day	16	2	38	12/20
	5	0.27	0.00	12.12	87.88	27,247	2636.27	16	2	38	12/20
	1	0.85	0.36	0.00	100	1,422,118	day	16	2	45	12/20
	2	0.22	0.00	3.43	96.57	551	312.06	16	2	45	13/20
70x20	3	0.14	0.00	10.20	89.80	7,380	941.37	16	2	45	13/20
	4	0.14	0.00	10.20	89.80	2,351	548.11	16	2	45	13/20
	5	0.19	0.00	10.49	89.51	3,536	893.07	16	2	45	13/20
	1	0.50	0.06	2.66	97.34	1,418,010	day	16	2	35	14/30
	2	1.06	0.07	3.14	96.86	1,096,231	day	16	2	37	18/30
80x30	3	0.23	0.13	8.37	91.63	1,772,804	day	16	2	37	18/30
	4	0.52	0.13	10.31	89.69	1,291,893	day	16	2	37	18/30
	5	0.26	0.00	10.31	89.69	181,005	43760.71	16	2	37	18/30
	1	0.81	0.39	1.75	98.25	467,034	day	16	2	37	21/40
	2	0.00	0.00	4.48	95.52	0	848.88	16	2	37	21/40
100x40	3	0.11	0.00	4.88	95.12	122,427	69552.32	16	2	37	21/40
	4	0.11	0.00	4.88	95.12	650,842	day	16	2	37	21/40
	5	0.06	0.04	4.88	95.12	646,893	day	16	2	37	21/40
average		0.93	0.09	8.71	91.29	845,884	29761.35	24	3	44	10

4.5.4 Summary of results

Our computational experiments can be summarized as follows. First, in terms of the static CSDS:

- the time indexed model we introduce significantly outperforms the one introduced by [15] in terms of the LP relaxation (i.e., the solution at the root node before branching), the optimality gap after terminating the branch and bound in CPLEX, and the cpu time.
- In Table 4.1, we are able to solve optimally, within few seconds, the benchmark instances that were not solved before in the literature.
- In Tables 4.2 and 4.2, we demonstrate the capabilities of this model in solving larger set of instances optimally within 2 hours of processing time as opposed to the model introduced by [15], where not of these instance were solve optimally.
- The time index model we introduced obtain on average a %0.06 optimality gap when CPLEX terminates after two hours whereas [15]'s model provides an average of %99.47. Most of the instances are solved optimally using our static CSDS model.

Second, in terms of the dynamic CSDS:

- the dynamic CSDS is shown to be more difficult to solve optimally than the static one due to the increment in the number of variables and constraints.
- the optimal solution was not found after one day of processing time for a few instances. However, the average optimality gap is still tight, i.e., %0.09.

4.5.5 Cost Parametric Analysis

As seen in this paper, the objective function in our CSDS models consists of two terms: tardiness and labor costs. In this section, we show via small instances how this cost changes, depending on the priority given to each term. We define two new parameters $\{a, b\}$, where a is the multiplier associated with the tardiness term and b is associated with the labor cost term. In all our previous experiments, we assumed $\{0.1, 1\}$. We demonstrate a variation of these parameters in Table 4.8. We test this parametric change on the instance $20x5$ with $q = \{1, 2, 3, 4, 5\}$. For each block we provide six combinations of the parameters $\{a, b\}$. Moreover, we additionally depict the change in cost between tardiness and labor in the last column of Table 4.8.

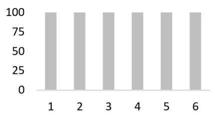
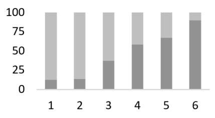
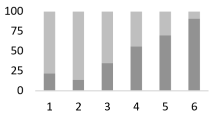
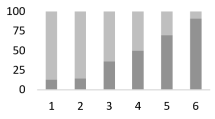
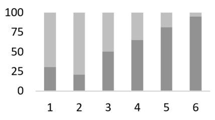
The first block of instances $20x5\,[q = 1]$ indicates no change in the cost composition between both objective terms. The reason is due to the looseness in the due dates. The remaining instances show an expected behavior of tardiness increments versus labor decrement as more weight is put on the tardiness while less is put on the labor. It is also important to mention the %LP gets tighter as more priority is given to the tardiness cost. In fact, we observe that some of the LP solutions are optimal.

4.6 Conclusion

In this paper, we have studied a cross-dock scheduling problem that integrates the assignment and sequencing of arriving containers with the selection of inbound dock doors to unload the containers. We have introduced a static IP model, $CSDS_1$, that outperforms the one introduced by Cao et al. [15], i.e., N_1, in terms of the %LP bounds, the speed at which it obtains optimal solutions, and the number of optimal solutions obtained.

Another contribution of this paper is the introduction of the dynamic CSDS with an associated MIP model that can solve most of the instances optimally, especially when due dates are scattered. These instances are a replication of the practice we have observed in a large crossdock on the East Coast of the US. To the best of our knowledge, the dynamic CSDS model introduced in this paper is the first dynamic model in the literature of cross-docking and can be extended for the machine scheduling and job selection environment as well. This paper opens a window for further research to be carried out for the development of decomposition methods to exploit the structure of the problem. Finally, we are considering the integration of other important goals, such as scheduling outgoing trailers in order to minimize the lateness of deliveries.

Table 4.8: Cost Parametric Analysis on the Dynamic CSDS with Scattered Due Dates

instance	{a,b}	%LP	%tardy	%labor	nodes	time	C_{max}	S_{max}	%shifts	selected	Depiction of cost change in %
20x5 [q=1]	{0.1,1}	6.05	0.00	100	12,899	27.80	24	3	33	2/5	
	{0.1,0.9}	6.05	0.00	100	3,343	10.80	24	3	33	2/5	
	{0.3,0.7}	6.05	0.00	100	2,678	6.69	24	3	33	2/5	
	{0.5,0.5}	6.05	0.00	100	15,084	25.36	24	3	33	2/5	
	{0.7,0.3}	6.05	0.00	100	15,286	27.31	24	3	33	2/5	
	{1,0.1}	6.05	0.00	100	12,936	19.66	24	3	33	2/5	
20x5 [q=2]	{0.1,1}	7.58	12.20	87.80	4,139	13.35	16	2	60	5/5	
	{0.1,0.9}	7.20	13.37	86.63	983	7.12	16	2	60	5/5	
	{0.3,0.7}	1.39	37.31	62.69	0	0.54	16	2	60	5/5	
	{0.5,0.5}	1.94	58.14	41.86	6	1.47	16	2	60	5/5	
	{0.7,0.3}	2.64	66.95	33.05	0	0.69	11	2	116	5/5	
	{1,0.1}	0.82	89.67	10.33	0	0.94	11	2	116	5/5	
20x5 [q=3]	{0.1,1}	7.72	21.47	78.53	1,714	8.83	24	3	33	3/5	
	{0.1,0.9}	8.03	13.37	86.63	30,557	62.91	16	2	60	5/5	
	{0.3,0.7}	2.17	34.88	65.12	0	0.63	14	2	69	5/5	
	{0.5,0.5}	1.85	55.56	44.44	0	0.60	16	2	60	5/5	
	{0.7,0.3}	1.68	69.83	30.17	0	0.57	14	2	80	5/5	
	{1,0.1}	0.51	90.84	9.16	0	0.74	14	2	80	5/5	
20x5 [q=4]	{0.1,1}	7.72	13.08	86.92	1,156	7.44	16	2	60	5/5	
	{0.1,0.9}	7.32	14.32	85.68	3,127	10.42	16	2	60	5/5	
	{0.3,0.7}	0.55	36.47	63.53	0	0.55	14	2	69	4/5	
	{0.5,0.5}	0.00	49.80	50.20	0	0.30	14	2	80	5/5	
	{0.7,0.3}	0.00	69.83	30.17	0	0.31	14	2	80	5/5	
	{1,0.1}	0.00	90.84	9.16	0	0.30	14	2	80	5/5	
20x5 [q=5]	{0.1,1}	7.39	30.50	69.50	4,950	15.56	16	2	9	4/5	
	{0.1,0.9}	7.60	20.96	79.04	1,066	6.99	16	2	60	5/5	
	{0.3,0.7}	1.73	50.56	49.44	0	0.61	16	2	60	5/5	
	{0.5,0.5}	0.00	64.98	35.02	0	0.37	12	2	93	5/5	
	{0.7,0.3}	0.00	81.24	18.76	0	0.53	12	2	93	5/5	
	{1,0.1}	0.00	94.89	5.11	0	0.53	12	2	93	5/5	

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis studied a fundamental class of optimization problems in cross-docking called *cross-dock door assignment problems* (CDAPs). We explored the structure of this basic problem theoretically and computationally, providing new mathematical formulations that allowed us to explore the structure further via decomposition techniques. For the last few years, the CDAP had been mostly formulated as a bilinear integer program causing difficulties in obtaining the optimal solutions for existing instances, and exploring the structure of the problem in depth. Hence, many researchers had resorted to heuristics. However, in this thesis we successfully introduced several new linear mixed-integer formulations for the CDAP. Not only did this allow for new instances to be solved optimally, but provided insights in utilizing the linear structure of these models. With this in mind, several linear, combinatorial and Lagrangean relaxations were introduced, explored, and where possible, obtained via well designed algorithms such as column generation and subgradient. We further were able to provide a methodology that provided lower and upper bounds on the CDAP, when solving the Lagrangian dual problem. A partial information was passed from the Lagrangian dual space to construct a complete feasible solution in the primal space, which resulted in feasible solutions that outperformed all existing heuristics in the literature. This should highlight one importance of all the lower bound procedures that we have developed as they generally should provide insights on the construction of feasible solutions. We also demonstrated how two bounds that are theoretically equivalent do not necessarily coincide in practice.

This thesis then extended the CDAP, only in its inbound side, by introducing sequencing and selection decisions, resulting in what we called the *Container Scheduling and cross-dock Door Selection* problem (CSDS). The integration of the assignments and sequencing decisions provided schedules, allowing for a practical realization of the CDAP in the short term horizon, with the objective of minimizing tardiness. The capacity constraints in the CDAP were no longer needed in the CSDS as they were replaced by the more realistic time constraints. Recall, the time constraints were reinterpreted by capacities on the dock doors for the CDAP. Then, came the introduction of the selection variables. We saw in the literature that the handling cost in most CDAPs was modeled with the assumption that it is the traveling distance between pair of doors at cross-dock facilities. We even saw that this handling cost had been interpreted as traveling time between doors, with unloading/loading times as well. So, what is "handling" after all? Is it time, distance, labor intensity, or congestion, or maybe a combination of some or all of these? The answer to this question could indeed rely on the specific cross-dock facility, and the operational handling cost it seeks to minimize. From our observation at the cross-dock facility in the USA, their concern when it comes to handling, was labor. In particular, the receivers who are in charge of unloading the containers. Therefore, in our efforts to integrate a CDAP into a CSDS, we also introduced the decision of selecting a dock door to be opened or closed as it reflected directly the associated labor cost. Finally, the integration of both scheduling and selection decisions would allow the cross-dock facility to trade off between their most important operational costs: tardiness and labor costs.

5.2 Future Work

The work of this thesis opens a new window of research for the CDAP and its extension CSDS. For instance, a flow based formulation with variables that characterize the amount of commodities between a pair of doors could be introduced for the CDAP to study its performance in terms of a branch and cut algorithm. Another aspect is in the utilization of column generation algorithm to be embedded into a branch and price algorithm to obtain the optimal solution for new instances that have not been solved yet, especially for the CDAP with loading/unloading times where the optimality gap is within $\%2$. Stabilizing CG, however, for efficiency purposes could be of a priority than a branch and price algorithm. We also observed that when removing the linking assignment constraints (3.10) and (3.11), and relaxing the integrality on the assignment variables, x and y

in (3.6) and (3.7) in the path-based formulation, $P1'$ in Chapter 3, we obtained a relaxed linear program that had a nice dual structure to explore. A heuristic algorithm could be easily developed to obtain feasible solutions on the dual problem, and hence, obtain lower bounds on $P1'$. We also noticed that many of the relaxed solutions obtained from either column generation or subgradient method, when used for constructing feasible solutions could impose some difficulty especially when the slackness of capacity constraints were tight. One way to mitigate this is by allowing for the construction of solutions that are not necessarily feasible (e.g., by enlarging the slackness) while penalizing their violations in the objective function. Such a strategy is known as *oscillation* and is known to be integrated in Tabu search for similar problems.

When it comes to the CSDS, the integration of sequencing and selection decisions was just one aspect of what we had observed in practice. Other important decisions include, for instance, the scheduling of departing trailers to minimize the lateness or late deliveries to customers. This could be integrated in our CSDS models if the departure time was fixed as seen in the postal industry as reported by Boysen et al. [12] and others. Alternatively, if the departure time was flexible, new decisions could be included to choose what and when a trailer may leave the cross-dock to meet delivery deadlines.

Bibliography

[1] I. Adiri, J. Bruno, E. Frostig, and A.R. Kan. Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, 26(7):679 – 696, 1989.

[2] D. Agustina, C.K.M. Lee, and R. Piplani. A review: Mathematical models for cross docking planning. *International Journal of Engineering Business Management*, 2(2):47 – 54, 2010.

[3] A. Ahlafçioglu, M. Bussieck, M. Esen, M. Guignard, J. Jagla, and A. Meeraus. Combining QCR and CHR for convex quadratic pure 0-1 programming problems with linear constraints. *Annals of Operations Research*, 199:33 – 49, 2012.

[4] B. Alidaee and H. Li. Parallel machine selection and job scheduling to minimize sum of machine holding cost, total machine time costs, and total tardiness costs. *IEEE Transactions on Automation Science and Engineering*, 11(1):294 – 301, 2014.

[5] A. Amini, R. Tavakkoli-Moghaddam, and A. Omidvar. Cross-docking truck scheduling with the arrival times for inbound trucks and the learning effect for unloading/loading processes. *Production & Manufacturing Research*, 2(1):784 – 804, 2014.

[6] H.M.B. Amor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167 – 1184, 2009.

[7] U.M. Apte and S. Viswanathan. Effective cross docking for improving distribution efficiencies. *International Journal of Logistics*, 3(3):291 – 302, 2000.

[8] G. Arnaout, E. Rodriguez-Velasquez, G. Rabadi, and R. Musa. Modeling cross-docking operations using discrete event simulation. In *Proceedings of the 6th International Workshop on Enterprise & Organizational Modeling and Simulation*, pages 113 – 120, 2010.

[9] J.J. Bartholdi and S.T. Hackman. Warehouse & distribution science: release 0.96. `http://www2.isye.gatech.edu/~jjb/wh/book/editions/wh-sci-0.96.pdf`, 2014. [Online; accessed 18-March-2017].

[10] N. Boysen. Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413 – 422, 2010.

[11] N. Boysen, M. Fliedner, and A. Scholl. Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, 32(1):135 – 161, 2010.

[12] N. Boysen, D. Briskorn, and M. Tschöke. Truck scheduling in cross-docking terminals with fixed outbound departures. *OR Spectrum*, 35(2):479 – 504, 2013.

[13] Y.A. Bozer and H.J. Carlo. Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions*, 40(11):1007 – 1018, 2008.

[14] P. Buijs, I.F.A. Vis, and H.J. Carlo. Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239(3):593 – 608, 2014.

[15] D. Cao, M. Chen, and G. Wan. Parallel machine selection and job scheduling to minimize machine cost and job tardiness. *Computers & Operations Research*, 32(8):1995 – 2012, 2005.

[16] F. Chen and K. Song. Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers & Operations Research*, 36(6):2066 – 2073, 2009.

[17] R. Chen, B. Fan, and G. Tang. Scheduling problems in cross docking. In *Combinatorial Optimization and Applications*, volume 5573, pages 421 – 429. Springer, Berlin, 2009.

[18] Z.L. Chen and W.B. Powell. Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1):78 – 94, 1999.

[19] A. Chmielewski, B. Naujoks, M. Janas, and U Clausen. Optimizing the door assignment in ltl-terminals. *Transportation Science*, 43(2):198 – 210, 2009.

[20] K.L. Choy, H.K.H. Chow, T.C. Poon, and G.T.S. Ho. Cross-dock job assignment problem in space-constrained industrial logistics distribution hubs with a single docking zone. *International Journal of Production Research*, 50(9):2439 – 2450, 2012.

[21] Y. Cohen and B. Keren. Trailer to door assignment in a synchronous cross-dock operation. *International Journal of Logistics Systems and Management*, 5(5):574 – 590, 2009.

[22] R.L. Cook, B. Gibson, and D. MacCurdy. A lean approach to cross docking. *Supply Chain Management Review*, 9(2):54 – 59, 2005.

[23] F. Enderer, C. Contardo, and I. Contreras. Integrating dock-door assignment and vehicle routing in cross-docking. Manuscript submitted for publication, 2015.

[24] L. Fanjul-Peyro and R. Ruiz. Scheduling unrelated parallel machines with optional machines and jobs selection. *Computers & Operations Research*, 39(7):1745 – 1753, 2012.

[25] G. Forger. UPS starts world's premiere cross-docking operation. *Modern Material Handling*, 36(8):36 – 38, 1995.

[26] S. Gelareh, R.N. Monemi, F. Semet, and G. Goncalves. A branch-and-cut algorithm for the truck dock assignment problem with operational time constraints. *European Journal of Operational Research*, 249(3):1144 – 1152, 2016.

[27] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82 – 114, 1974.

[28] M. Guignard, P.M. Hahn, A.A Pessoa, and D.C da Silva. Algorithms for the crossdock door assignment problem. In *Proceedings of the Fourth International Workshop on Model-Based Metaheuristics*. Brazil., 2012.

[29] P. Hahn, J.M. Smith, and Y.-R. Zhu. The multi-story space assignment problem. *Annals of Operations Research*, 179:77 – 103, 2010.

[30] J.C. Ho and Y.L. Chang. Heuristics for minimizing mean tardiness for m parallel machines. *Naval Research Logistics*, 38(3):367 – 381, 1991.

[31] E. Kinnear. Is there any magic in cross-docking? *Supply Chain Management: An International Journal*, 2(2):49 – 52, 1997.

[32] A.L. Ladier and G. Alpan. Cross-docking operations: Current research versus industry practice. *Omega*, 62:145 – 162, 2016.

[33] C.Y. Lee, L. Lei, and M. Pinedo. Current trends in deterministic scheduling. *Annals of Operations Research*, 70:1 – 41, 1997.

[34] Y. Li, P. Pardalos, K. Ramakrishnan, and M. Resende. Lower bounds for the quadratic assignment problem. *Annals of Operations Research*, 50(1):387 – 410, 1994.

[35] T.W. Liao, P.J. Egbelu, and P.C. Chang. Simultaneous dock assignment and sequencing of inbound trucks under a fixed outbound truck schedule in multi-door cross docking operations. *International Journal of Production Economics*, 141(1):212 – 229, 2013.

[36] Z. Liu and E. Sanlaville. Stochastic scheduling with variable profile and precedence constraints. *SIAM Journal on Computing*, 26(1):173 – 187, 1997.

[37] G. Luo and J.S. Noble. An integrated model for crossdock operations including staging. *International Journal of Production Research*, 50(9):2451 – 2464, 2012.

[38] Y. Ma, C. Chu, and C. Zuo. A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58(2):199 – 211, 2010.

[39] S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research*, 123(2):325 – 332, 2000.

[40] Z. Miao, A. Lim, and H. Ma. Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research*, 192(1):105 – 115, 2009.

[41] M. Napolitano. Cross dock fuels growth at dots. *Logistics Management*, 50(2):30 – 34, 2011.

[42] W. Nassief, I. Contreras, and R. As'ad. A mixed-integer programming formulation and lagrangean relaxation for the cross-dock door assignment problem. *International Journal of Production Research*, 54:494 – 508, 2016.

[43] W. Nassief, I. Contreras, M. Guignard, P. Hahn, and B. Jaumard. The container scheduling and cross-dock door selection problem. Manuscript submitted for publication, 2017.

[44] W. Nassief, I. Contreras, and B. Jaumard. A comparison of formulations and relaxations for the cross-dock door assignment problems. Manuscript submitted for publication, 2017.

[45] Y. Oh, H. Hwang, C.N. Cha, and S. Lee. A dock-door assignment problem for the korean mail distribution center. *Computers & Industrial Engineering*, 51(2):288 – 296, 2006.

[46] K.E. Peck. *Operational Analysis of Freight Terminals Handling Less than Container Load Shipments*. PhD thesis, University of Illinois, Urbana-Champaign, IL, United States, 1983.

[47] M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Springer, New York, 2012.

[48] C.R. Rosales, M.J. Fry, and R. Radhakrishnan. Transfreight reduces costs and balances workload at georgetown crossdock. *Interfaces*, 39(4):316 – 328, 2009.

[49] A. Ross and V. Jayaraman. An evaluation of new heuristics for the location of cross-docks distribution centers in supply chain network design. *Computers & Industrial Engineering*, 55 (1):64 – 79, 2008.

[50] Saddle Creek Corporation. 2011 Cross-Docking Trends Report. `http://www.thomasnet.com/white-papers/abstract/101631/2011-crossdocking-trends-report.html`, 2011. [Online; accessed 14-March-2017].

[51] E. Sanlaville and G. Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795 – 811, 1998.

[52] H.D. Sherali and W.P. Adams. Reformulation–linearization techniques for discrete optimization problems. In *Handbook of Combinatorial Optimization*, pages 2849 – 2896. Springer, New York, 2013.

[53] A. Shuib and W.N.A.W.A. Fatthi. A review on quantitative approaches for dock door assignment in cross-docking. *International Journal on Advanced Science, Engineering and Information Technology*, 2(5):30 – 34, 2012.

[54] G. Stalk, P. Evans, and L.E. Shulman. *Competing on capabilities: the new rules of corporate strategy*, volume 63. Harvard Business Review, 1992.

[55] L.Y. Tsui and C.-H. Chang. A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering*, 19(1):309 – 312, 1990.

[56] L.Y. Tsui and C.-H. Chang. An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering*, 23(1):283 – 286, 1992.

[57] Y. Unlu and S.J. Mason. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58 (4):785 – 800, 2010.

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

[59] J.F. Wang and A. Regan. Real-time trailer scheduling for crossdock operations. *Transportation Journal*, pages 5 – 20, 2008.

[60] C.E. Witt. Crossdocking: Concepts demand choice. *Material Handling Engineering*, 53(7): 44 – 49, 1998.

[61] T. Zhang, G.K.D. Saharidis, S. Theofanis, and M. Boile. Scheduling of inbound and outbound trucks at cross-docks: Modeling and analysis. *Transportation Research Record: Journal of the Transportation Research Board*, 2162:9 – 16, 2010.

[62] Y.-R. Zhu, P. Hahn, Y. Liu, and M. Guignard. New approach for the cross-dock door assignment problem. In *Anais do XLI Simposio de Pesquisa Operacional, Porto Seguro, Bahia, Brazil*, 2009.