Optimization Models and Methods for Storage Yard Operations in Maritime Container Terminals

by

Virgile Galle

M.S. Ecole Centrale Paris (2015)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

Author .....
Sloan School of Management
January 12, 2018

Certified by .....
Cynthia Barnhart
Chancellor
Ford Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by .....
Patrick Jaillet
Co-director, Operations Research Center
Dugald C. Jackson Professor of Electrical Engineering & Computer Science
Thesis Supervisor

Accepted by .....
Dimitris Bertsimas
Co-director, Operations Research Center

Optimization Models and Methods for Storage Yard Operations in Maritime Container Terminals

by

Virgile Galle

Submitted to the Sloan School of Management
on January 12, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Container terminals, where containers are transferred between different modes of transportation both on the seaside and landside, are crucial links in intercontinental supply chains. The rapid growth of container shipping and the increasing competitive pressure to lower rates result in demand for higher productivity.

In this thesis, we design new models and methods for the combinatorial optimization problems representing storage yard operations in maritime container terminals. The goal is to increase the efficiency of yard cranes by decreasing unproductive container moves (also called relocations). We consider three problems with applicability to real-time operations.

First, we study the *container relocation problem* that involves finding a sequence of container moves that minimizes the number of relocations needed to retrieve all containers, while respecting a given order of retrieval. We propose a new binary integer program model, perform an asymptotic average case analysis, and show that our methods can apply to other storage systems where stacking occurs.

Second, we relax the assumption that the full retrieval order of containers is known in advance and study the *stochastic container relocation problem*. We introduce a new model, compare it with an existing one, and develop two new algorithms for both models based on decision trees and new heuristics. We show that techniques in this chapter apply more generally to finite horizon stochastic optimization problems with bounded cost functions.

Third, we consider the integrated container relocation problem and *yard crane scheduling problem* to find an optimal sequence of scheduled crane moves that perform the required container movements. Taking into account practical constraints, we present a new model, propose a binary integer program using a network flow-type formulation, and design an efficient heuristic procedure for real-time operations based on properties of our mathematical formulation. We relate this problem to pick-up and delivery problems with a single vehicle and capacities at every node.

In all three chapters, the efficiency of all our algorithms are shown through extensive computational experiments on available problem instances from the literature

and/or on real data.

Thesis Supervisor: Cynthia Barnhart
Title: Chancellor
Title: Ford Professor of Civil and Environmental Engineering

Thesis Supervisor: Patrick Jaillet
Title: Co-director, Operations Research Center
Title: Dugald C. Jackson Professor of Electrical Engineering & Computer Science

Acknowledgments

"Tell me and I forget, teach me and I may remember, involve me and I learn."

B. Franklin

First and foremost, I would like to thank my advisors Cynthia Barnhart and Patrick Jaillet for their guidance and support during my stay at MIT. Cindy and Patrick are not only incredible and brilliant academic advisors, they have also been real life mentors. I would like to thank them for their patience, their flexibility, their advice, and their kindness. In addition, they have given me many opportunities to collaborate with brilliant people, to attend several conferences, and to intern for two summers at great companies. I will always remember our meetings in LIDS, in 10-200, on Skype, and especially the first presentations that I gave them. They really taught me the way to convey a message properly, in research and in life.

I would also like to thank my doctoral committee members Vahideh Manshadi and Juan Pablo Vielma. Vahideh has been a real role model as well as a mentor for me. She has helped me countless times during these four years and a half. Juan Pablo is one of the best professors I had at MIT. His expertise and insights have been of incredible value for my research. I would also like to acknowledge an anonymous donor without whom this work would not have been possible. I would also like to thank Laura Rose and Andrew Carvalho who always offered their help over the years as well as all the ORC faculty members. I am also grateful to Setareh Borjian for being a colleague and a friend my first two years at MIT.

My PhD has also been an amazing adventure with all the friends I made at MIT. I am grateful to all my friend from the ORC crew as well as the ORC honorary members for the memories at MIT and on our different trips in the U.S. (San Francisco, Philadelphia, Ski trips, Retreat in Maine, New Orleans, Houston, Austin, Las Vegas, Cape Cod, ...). Special thanks to my roommates, past and present, Andrew L., Arthur F., Zachary O., Mathieu D., Mariapaola T., Ludovica R. and Cécile C. (at 250 Western and Sidpac); to my very good friends and fellow INFORMS officers

Daniel Schonfeld and Joey H. (may JLVD live forever); to the French crew Pierre B., Alexandre S., Ali A., Claire-Marine W., Anne C., Sébastien M., Max Burq, and Florian F.; to the best match maker Jean Pauphilet; to the ORC soccer team with Alexander R., Kevin R., and Nikita K.; to Daniel Smilkov and Maxime C. for welcoming me to the US; and thanks to Daw-sen H., Anna P., Charles T., Stefano T., David H. (alias Scott), Nishanth M., Rim H., and Max Biggs for all the great times together.

I would also like to express my gratitude to some friends I met during my "trips" to Stanford: Claire D., Matthias C. and Victor L. Finally, I cannot forget my friends back in France: Jean-Baptiste P., Mathieu D. and Évrard B.. Special thanks to Pierrick Piette for coming twice to Boston and for being a supportive and amazing friend.

My family has always been there for me and their support has been very important for me. Thanks to my mum Geneviève, my dad Jean-Loïc and my brothers, Aurèle and Hector. I would also like to thank my godmother Catherine, my godfather Yann and all my aunts, uncles and cousins. My gratitude also goes to my future parents-in-law, Carole and Olivier, and my brother-in-law, Julien.

Last but not least, my greatest thanks goes to the love of my life, Sophie Trastour. During this journey, I have relied on her unconditional support and care. She has been my greatest motivator and friend throughout. Her perseverance, calm and brilliant mind have helped me get through the toughest times. Thanks to her, I got to know the greatest Léa and Rose, my other loves. This accomplishment also belongs to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"A simple calculation shows that there are enough containers on the planet to build more than two 8-foot-high walls around the equator" ([27])

This chapter provides a general overview of container terminals and their operations. Section 1.1 describes the continuously growing importance of container terminals in today's international trade system. Section 1.2 introduces the typical handling equipment, layout and new technologies of container terminals. Section 1.3 describes the wide range of optimization problems arising in container terminals. Section 1.4 presents the common notations and some general mathematical background for subsequent chapters of the thesis. Section 1.5 details the structure of the thesis along with the main contributions of each chapter.

1.1 Container Terminals

The birth of container shipping is traditionally considered to be April 1956 when Malcolm McLean transported 58 containers from Newark to Houston (see Figure 1-1). Since then, the container shipping industry has continuously grown. In 2017, the largest container ship (see Figure 1-2) can transport more than 21,400 Twenty-Foot Equivalent Units (TEUs). Today, the term *containerization* refers to the system of intermodal freight transport using containers. In such systems, the dimension of

containers (20 foot equivalent units (1 TEU), 40 foot equivalent units (2 TEUs), 45 foot equivalent units (high-cubes)) needed to be standardized to ease their use by various means of transportation (ships, truck, trains,...).

Figure 1-1: Malcolm McLean's refitted oil tanker carrying the first container shipment in April 1956 (source: Commons wikimedia).

Figure 1-2: Largest container ship in 2017, operated by Orient Overseas Container Line (source: occl.com).

This trend of containerization, confirmed by the evolution of seaborne trade carried by container ships, is presented in a study conducted beginning in 2017. Figure 1-3 shows that the weight of goods carried via containers has multiplied by more than 16 from 1980 to 2016. In addition, Alphaliner.com estimated the number of TEUs in the global containership fleet on January 1^{st} 2017 to be around 23.4 million. Naturally, the major owners of these containers are also major ship operators. As of

October 30 2017, APM-Maersk, Mediterranean Shg Co and CMA CGM Group were the top 3 owners in both categories.

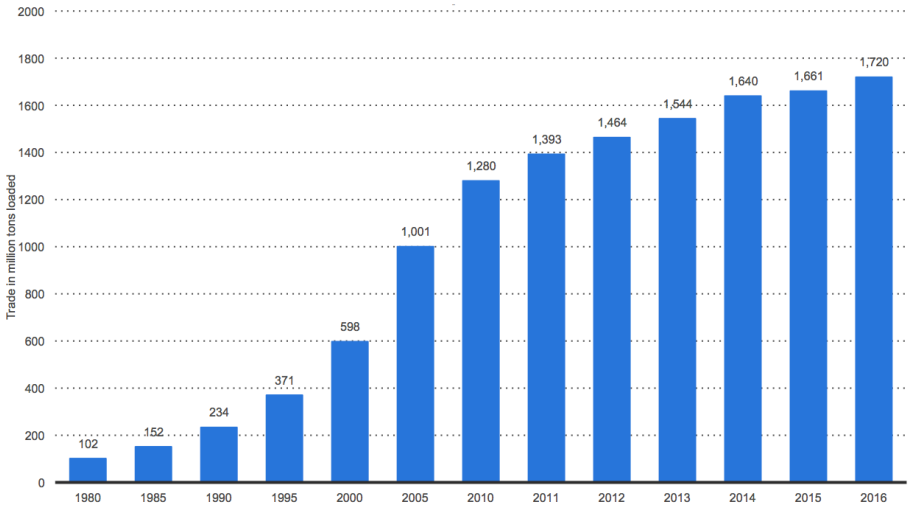


Figure 1-3: International seaborne trade carried by container ships from 1980 to 2016 in million tons loaded (source: UNCTAD, Clarkson Research Services).

With the rapid growth of container shipping, container terminals have flourished to become crucial links in intercontinental supply chains. Indeed, this is where containers are transferred between the different modes of transportation. Figure 1-4 identifies the major global terminal operators in 2015 based on market share.

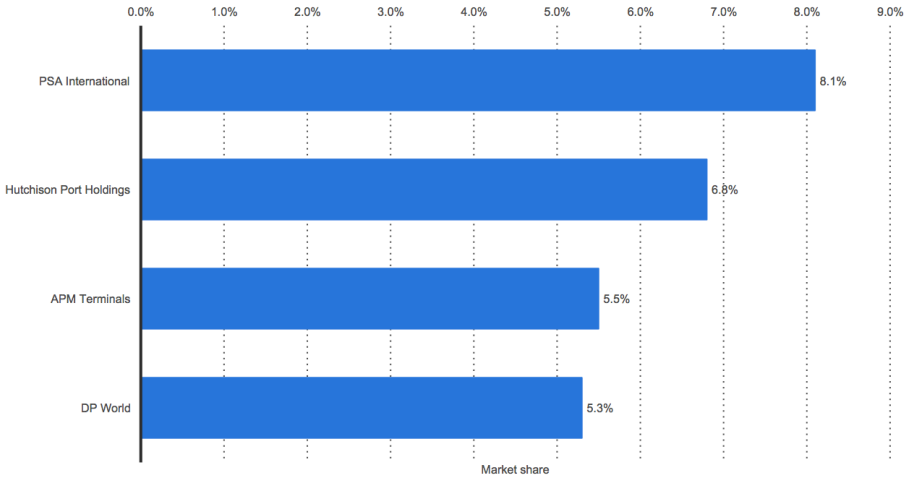


Figure 1-4: Market share of major terminal operators worldwide as of mid-year 2015 (source: Drewry).

Finally, Figure 1-5 shows a map of the top 50 largest terminals in terms of through-

put in 2012. Most large terminals are located in Eastern Asia, with the other ports widely spread around the globe, demonstrating the global impact of containerization.

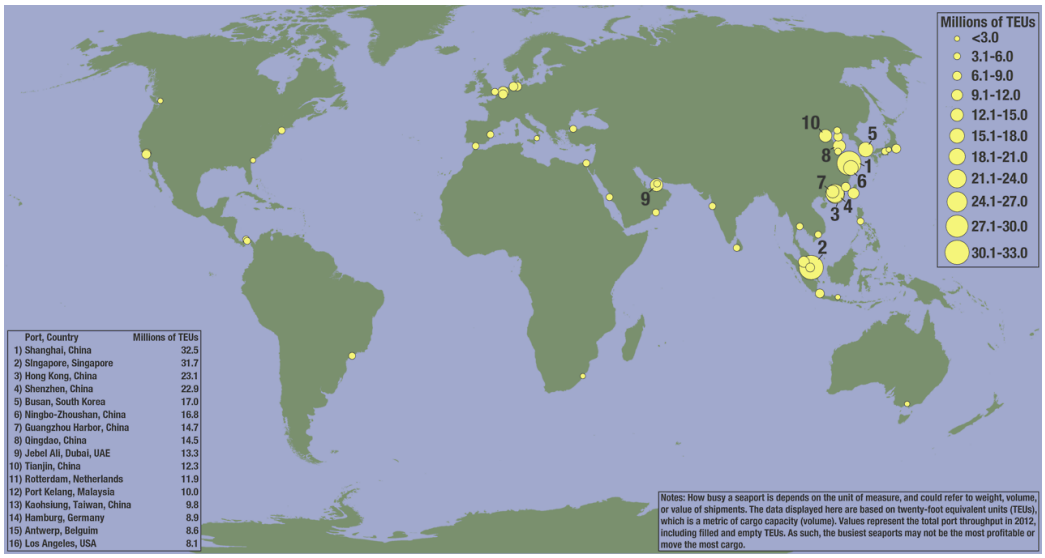


Figure 1-5: The world's top 50 containers in 2012 in terms of throughput (source: World Shipping Council).

1.2 Handling Equipment, Layout and New Technologies

The typical handling equipment at container terminals involves *Quay Cranes* (QC); *Yard Cranes* which can be a single rubber-tired gantry crane (RTG), a single rail-mounted gantry crane (RMG), double RMGs, twin RMGs and triple RMGs; *Internal vehicles* including yard trucks, straddle carriers, chassis, automated guided vehicles (AGV) and land cranes (used to load trains); and *External modes of transportation* such as external trucks or trains.

Figure 1-6 presents the typical layout of a container terminal with the three main sections of a container terminal: 1. *the Sea-side* including the Vessel, the Quay and the Internal transport areas; 2. *the Storage Yard*; and 3. *the Land-side* including external transport areas and the gate. Each piece of equipment operates in one or several sections of a container terminal (see Figure 1-6).

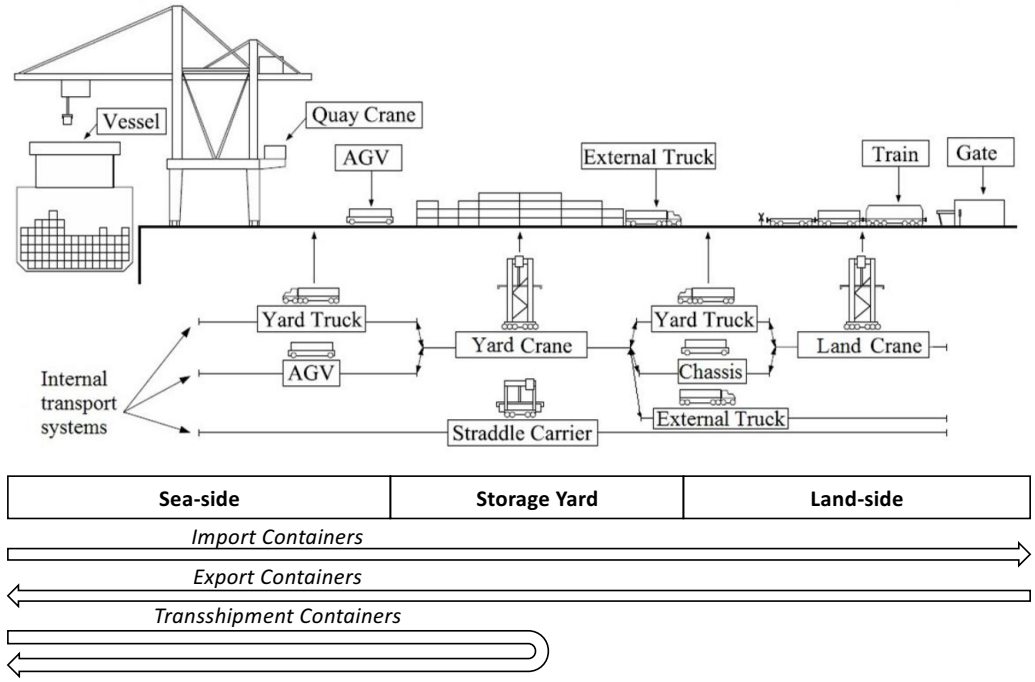


Figure 1-6: Schematic representation of a container terminal (adapted from [27]).

A container can belong to one of three types: *Import containers* arrive from the sea-side and leave on inland transport modes; *export containers* arrive from inland transport modes and leave by the sea-side on vessels; and *transshipment containers* arrive and leave on the sea-side. Typically, empty, refrigerated (or reefer) and hazardous containers are usually assigned to specific areas within the yard.

The choice of layout and equipment highly impact the productivity of the terminal and highly depend on the proportion of import, export, and transshipment containers, as well as the overall target storage capacity of the terminal. For instance, straddle carriers provide better flexibility but a lower storage capacity than yard cranes. Several studies, such as the most recent one in [37], present a more detailed list of handling equipment and terminal layouts and evaluate their impact on the overall terminal performance under different metrics.

Finally, new technologies constantly impact all parts of container terminal operations. On the sea-side, a new generation of fully automated quay cranes have been developed in the past 5 years with double or triple lifting trolley and shuttles to perform faster horizontal movements of containers. Concerning internal transport, the

number of automated vehicles has increased significantly, enhancing the use of GPS and RFID technologies to create coordination and track containers and resources. In the storage yard, new cranes with handling and overpassing capabilities have been engineered. Finally, many innovative solutions have focused on designing new layouts and stacking systems based on innovations in warehousing, such as rack-based compact storage or overhead grid rail systems, with the most famous example being the ultra-high container warehouse of Ez-Indus in South Korea.

1.3 Operations Research Models

Since 2008, researchers have introduced and studied many operations research models to tackle various problems at container terminals. Gharehgozli et al. [27] report 177 papers published since 2008 that consider optimization problems in container terminals. The following list summarizes the main models found in the literature classified by sections of the terminal. For extensive literature reviews on general container terminal operations, we refer the reader to [27, 64].

Sea-side. There are four main problems related to sea-side operations. Bierwirth and Miesel [2] present an extensive survey of papers studying the following problems before 2010.

The *container stowage problem (CSP)* (for a recent example, see [16]) is concerned with the placement of a container at a ship slot to minimize the port stay times of ships, ensure stability, obey stress operating limits of the ships, and maximize QC utilization. Container characteristics such as weight, size, port of unloading, and type (reefer or hazardous) are typically taken into account as constraints.

The *berth allocation problem (BAP)* (for a recent example, see[78]) considers the minimization of ship waiting and handling times given spatial (discrete vs continuous berths), temporal (static or dynamic), and handling time (dependent on berthing position, QC assignment, and/or QC schedules vs. fixed) constraints.

The *quay crane assignment problem (QCAP)* aims at reducing the number of QC

setups and QC travel times to maximize crane productivity. In practice, QCAP is solved by greedy rules. This problem has mostly been considered jointly with the BAP in papers such as in [30].

The *quay crane scheduling problem (QCSP)* (for a recent example, see [48]) tackles the design of optimal schedules for QCs to maximize throughput, and minimize ship handling time while satisfying constraints such as crane crossovers, minimum distance between cranes, and unloading before loading.

Finally, more recently, works similar to [13] have integrated the BAP, QCAP and QCSP to provide more globally optimal solutions.

Land-side.

Internal transportation problems include determining shortest-time routes, sequencing requests, dispatching vehicles in real-time, and sizing the fleet of internal vehicles (see [8]).

Gate operations planning problems primarily consist of truck appointment system design (also called time window management), train loading and unloading - as well as congestion reduction at the gates - by analyzing queuing models or through simulation.

Storage yard. The literature has historically been divided between yard crane operations planning (see YSCP) and container relocations minimization (see CRP, SP, DCRP and PMP). A detailed discussion of these problems is provided in Chapter 2.

The *yard crane scheduling problem (YCSP)* is concerned with sequencing storage and retrieval operations - without considering relocations - to minimize makespan or average vehicle job waiting time (or delay) or to maximize crane utilization.

The *container relocation problem (CRP)* (also called block relocation problem, BRP) is concerned with finding a sequence of moves of containers that minimizes the number of relocations needed to retrieve all containers, while respecting a given order of retrieval.

The *stacking problem (SP)* aims at properly locating incoming containers such that the future handling effort (relocation or pre-marshalling as defined below) are decreased significantly.

The *dynamic container relocation problem (DCRP)* results from the combination of the CRP and the SP.

The *pre-marshalling problem (PMP)* identifies possibilities to avoid future relocations by pre-marshalling containers. Pre-marshalling corresponds to re-positioning containers so that a minimum number of relocations are needed when containers are loaded onto the ships. This problem applies especially in the case of export and transshipment containers when the ship's stowage plans are known in advance.

1.4 Notations and Mathematical Background

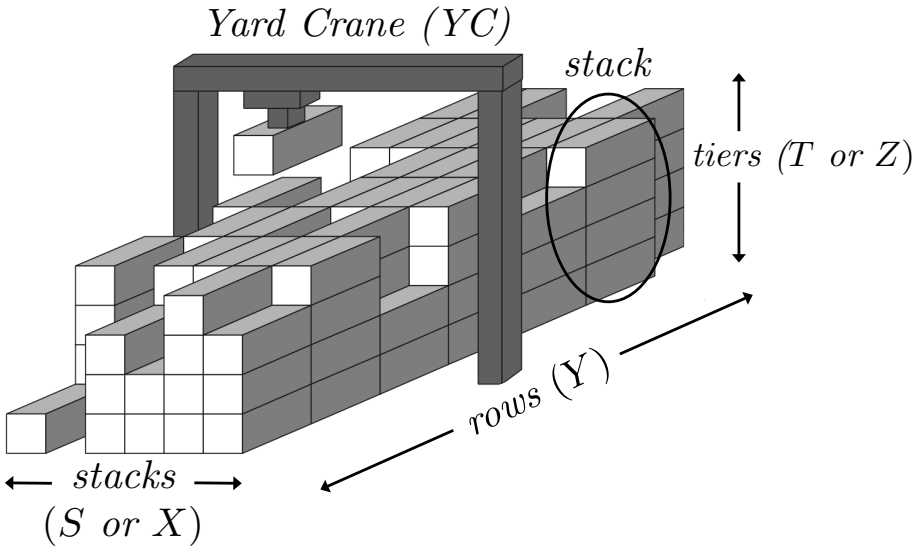


Figure 1-7: Geometry and notations of a container block.

Due to the lack of space, containers are stacked on the top of each other creating blocks of containers as shown in Figure 1-7. A block consists of S (or X) *stacks*, Y *rows* and T (or Z) *tiers* (see Figure 1-7) and we assume that this block is served by a single *yard crane* (YC). In several problems, only one row is considered. A *configuration* refers to a two-dimensional array representing a single row. In addition, we denote

by C the number of containers initially in a given configuration or block.

This thesis draws on several analyses enabled by recent advancements in mathematical research. Chapters 3 and 5 introduce binary integer programs. The efficacy of integer programming has recently been boosted by the great improvement of solvers such as Gurobi and the dramatic speed-ups of computational processors. Junger et al. [36] present a recent review on integer programming techniques. Chapter 4 and 5 tackle stochastic optimization problems which can be formulated as dynamic programming models. Related to this topic, Bertsekas [1] and Sennott [62] provide a general review of techniques on finite horizon dynamic programming as well as some concepts from approximate dynamic programming.

1.5 Overview and Contributions of the Thesis

Chapter 2: literature review. In this chapter, we present an extensive literature review of the main optimization problems in container terminal storage yards. We first present a thorough literature review on the container relocation problem and its variants as well as the yard crane scheduling problem. Then, we provide a general literature review on the storage problem and the pre-marshalling problem.

Chapter 3: the container relocation problem. This chapter focuses on the restricted container relocation problem enforcing that only containers blocking the target container can be relocated. The first section of this chapter is based on our published paper *A New Binary Formulation of the Restricted Container Relocation Problem Based on a Binary Encoding of Configurations* [22]. First, we improve upon and enhance an existing binary encoding and using it, formulate the restricted CRP as a binary integer programming problem in which we exploit structural properties of the optimal solution. This integer programming formulation reduces significantly the number of variables and constraints compared to existing formulations. Its efficiency is shown through computational results on small and medium sized instances taken from the literature. Subsequently, the second section is based on our published paper

An Average-Case Asymptotic Analysis of the Container Relocation Problem [25]. We focus on average case analysis of the CRP when the number of stacks grows asymptotically. We show that the expected minimum number of relocations converges to a simple and intuitive lower bound for which we give an analytical formula.

While this is not developed in the thesis, we mention that the author also developed an A* based algorithm presented in [4]. Finally, we developed two Matlab GUI interfaces. The first interface is a practical decision tool of potential interest for practitioners and can use solutions both from this chapter and Chapter 4. The second one could be used to test and compare human abilities with different algorithms. Both user interfaces are available at https://github.com/vgalle/CRP_GUIs.

Chapter 4: the stochastic container relocation problem. In the CRP, the assumption of knowing the full retrieval order of containers is particularly unrealistic in real operations. This chapter studies the stochastic CRP (SCRP), which relaxes this assumption. It is based on our submitted paper *The Stochastic Container Relocation Problem* [24]. A new multistage stochastic model, called the batch model, is introduced, motivated, and compared with an existing model (the online model). The two main contributions are an optimal algorithm called Pruning-Best-First-Search (PBFS) and a randomized approximate algorithm called PBFS-Approximate with a bounded average error. Both algorithms, applicable in the batch and online models, are based on a new family of lower bounds for which we show some theoretical properties. Moreover, we introduce two new heuristics outperforming the best existing heuristics. Algorithms, bounds and heuristics are tested in an extensive computational section. Finally, based on strong computational evidence, we conjecture the optimality of the "leveling" heuristic in a special "no information" case, for which any of the remaining containers, at any retrieval stage, is equally likely to be retrieved next.

Chapter 5: the yard crane scheduling problem with relocations. In the previous chapters, some practical characteristics such as stacking, the third dimen-

sion of the block, the actual travel time of the crane and limited flexibility of the order in which service can occur are not taken into account. This chapter considers a more practical model and introduces a novel optimization problem resulting from the integration of the yard crane scheduling problem and the container relocation problem. The work in this chapter is based on our working paper *Yard Crane Scheduling for Container Storage, Retrieval, and Relocation* [23]. This chapter is the first work to consider a general model that integrates the challenges of these two problems by simultaneously scheduling storage, retrieval and relocations requests and deciding on storage and relocation positions. We formulate this problem as an integer program that jointly optimizes current crane travel time and future relocations. Based on the structure of the proposed formulation, we propose a heuristic based on the LP relaxation of subproblems embedded in a local search scheme. Finally, we show the value of our solutions on both simulated instances as well as real data from a port terminal.

Chapter 6: concluding remarks. This chapter presents the main conclusions drawn from the thesis and suggests several directions for future research.

Chapter 2

Literature Review

General reviews and classification surveys of the existing literature on the container relocation problem, the yard crane scheduling problem and other related problems such as stacking or pre-marshalling problems can be found in [27, 7, 49, 64, 65].

2.1 The Container Relocation Problem and its Variants

2.1.1 The Container Relocation Problem

Due to limited space in the storage area of maritime ports, containers are stacked on top of each other. The resulting stacks create rows of containers as shown in 1-7. If a container that needs to be retrieved (target container) is not located at the top and is covered by other containers, blocking containers must be relocated. As a result, during the retrieval process, the yard cranes perform one or more relocation moves. Such relocations (also called reshuffles) are costly for the port operators and result in delays in the retrieval process. The container relocation problem (CRP) (also known as the block relocation problem) addresses this challenge by minimizing the number of relocations. The CRP applies to a broad range of two-dimensional storage systems involving containers, boxes, pallets or steel plates.

The CRP with the classical assumptions described in detail in Section 3.2 is re-

ferred to as *static and full information*: *static* because no new containers arrive during the retrieval process and *full information* because we know the full retrieval order at the beginning of the retrieval process. Finally, we mention that the *restricted CRP* assumes that only containers blocking the target container can be moved.

Under these assumptions, the problem was first formulated in [38] in a dynamic programming model. It has been shown in [10] that both the restricted and unrestricted CRP are $\mathcal{NP}$-hard.The solution approaches developed in literature on the CRP can be partitioned into three: integer programming approaches, other exact approaches such as branch-and-bound or A* algorithm and finally heuristic solution approaches.

Integer programming approaches

Wan et al. [74] formulate one of the first integer programming models for the CRP and develop an IP-based heuristic capable of obtaining near-optimal solutions. Caserta et al. [10] propose another intuitive formulation of the problem, called BRP-II, as well as an efficient heuristic. Tang et al. [67] propose a very similar formulation with fewer variables than in BRP-II, present heuristics and a worst case analysis. Expósito-Izquierdo et al. [20] correct BRP-II and rename their new formulation BRP-II*. Eskandari and Azari [18] also correct BRP-II and propose an improved formulation called BRP2ci by adding valid inequalities. Zehendner et al. [81] correct and improve BRP-II to get formulation BRP-II-A by removing some variables, tightening some constraints, introducing a new upper bound, and applying a preprocessing step to fix several variables. The two latter formulations being the most recent ones, we compare our new formulation to these state-of-the-art solutions in Section 3.3. As we mentioned, these are both improved corrections of BRP-II in [10], but they differ in the nature of added cuts as well as the preprocessing step in [81]. In addition, the computational results provided by both studies differ. While Zehendner et al. [81] give results for BRP-II-A on average over set of instances, Eskandari and Azari [18] present results only on a small subset of instances for BRP2ci, making their comparison difficult. Consequently, we show through our experiments that our new

formulation outperforms BRP2ci on available instances from [18] as well as BRP-II-A on average over sets of instances. Finally, Caserta et al.[10] and Petering and Hussein [58] develop formulations for the unrestricted CRP, but both are unable to solve small-sized instances efficiently.

Other exact approaches

Like for most multi-period combinatorial optimization problems, the previous IP formulations require variables with many indices. Therefore, even though the number of variables and constraints are polynomial in the size of the problem, these formulations become too large to even fit in memory of actual solvers in the case of real-sized problems. To bypass this issue, a recent trend has been to look at more efficient ways to explore the branch-and-bound tree or even decrease its size using the structural properties of the problem. Ünlüyurt and Aydın [71] and Expósito-Izquierdo et al. [20] suggest two branch-and-bound approaches with several heuristics based on this idea. Another solution using the A^* algorithm is explored in [87], and built upon in [4, 66]. Zehendner and Feillet [83] present another solution using branch-and-price and Ku and Arthanari [42] introduce another solution approach based on the abstraction method. More recently, Tricoire et al. [70] use an improved B&B to solve the unrestricted problem.

Heuristics

As both the restrcited and unrestricted CRP are $\mathcal{NP}$-hard (see [10]), an alternative approach is to use quick and efficient heuristics providing sub-optimal solutions such as in [21, 35]. Caserta et al. [10] introduce a "MinMax" policy that is defined and generalized later in the thesis. Wu and Ting [76] propose a beam search heuristic, and Wu and Ting [77] develop the Group Assignment Heuristic (GAH). Tricoire et al. [70] develops a rake search heuristic. Finally, we mention lower bounds for the CRP are developed in [38, 87, 66].

Available instances

To evaluate the efficiency of these methods, several sets of instances have been used. The most common one appears in [9] and is used in [10, 87, 58, 5, 18, 20, 81]. In these instances, T and S range from 3 to 10, C is taken to be $(T-2)S$ with $T-2$ containers per stack resulting in 21 classes of problem. With 40 instances per class, this set contains a total of 840 instances. We use these instances in Section 3.3.3. Instances from [47] consider multiple rows and are used to test heuristics. Zhu et al. [87] introduce both instances with distinct and non-distinct priorities.

2.1.2 The Stochastic Container Relocation Problem

As it was previously mentioned, the assumption of full information on the retrieval order is unrealistic given that arrival times of external trucks at the terminal are generally unpredictable due to uncertain conditions. Nevertheless, new technology advancements such as truck appointment systems (TAS's) and GPS tracking can help predict relative truck arrival times. Thus, although the exact retrieval order might not be known, some information on trucks' arrival times might be available, which motivates the introduction of a stochastic version of the CRP.

A common assumption is that, for each container, there is a time window in which a truck driver will arrive to retrieve it. We refer to a *batch of containers* as the set of containers stacked in the same row and with the same arrival time window. This information can be either inferred using machine learning algorithms, not yet much discussed in the literature or obtained by using the appointment time windows in a TAS, which has gained attention over the last decade. The first TAS was implemented by Hong Kong International Terminals (HIT) in 1997. It uses 30-minute time slots, where trucks can register (see [53]). Another TAS was introduced in New Zealand in 2007. Two other studies (see [31, 52]) evaluate the impact of TAS, in reducing truck idling time by increasing on-time ratio. More recent information can be found in [59, 3].

On the modeling side, Zehendner and Feillet [82] formulate an IP to get the

optimal number of slots a TAS should offer for each batch. Very few studies have tackled the SCRP, also referred to as CRP with Time Windows. This problem was first modeled by Zhao and Goodchild [86]. In their original model, each container is assigned to a batch. Batches of containers are ordered such that all containers in a batch must be retrieved before any containers from a later batch are retrieved. Furthermore, the relative retrieval order of containers within a given batch is assumed to be a random permutation. In Chapter 4, we will refer to the model from [86] as the *online model*. In Section 4.2, we discuss in more detail how this model assumes information is revealed. For the online model, Zhao and Goodchild [86] develop a myopic heuristic (called RDH) and study, in different settings with two or more groups, the value of information using RDH. They conclude that a small improvement in the information system reduces the number of relocations significantly. Asperen et al. [72] use a simulation tool to evaluate the effect of a TAS on many statistics including the ratio of relocations to retrievals. Their decision rules are based on several heuristics including the "leveling," random or "traveling distance" heuristics. More recently, Ku and Arthanari [43] also use the online model. They formulate the SCRP under the online model as a finite horizon dynamic programming problem, and suggest a decision tree scheme to solve it optimally. They also introduce a new heuristic called expected reshuffling index (ERI), which outperforms RDH, and they perform computational experiments based on available test instances. In Chapter 4, we refer frequently to this work, use some of their techniques, as well as their available test instances to evaluate our algorithms.

In another recent study related to the SCRP, Zehendner et al. [84] study the *online* container relocation problem, which corresponds to an adversarial model. They prove that the number of relocations performed by the leveling policy can be upper-bounded by a linear function of the number of blocking containers and provide a tight competitive ratio for this policy.

2.1.3 The Dynamic Container Relocation Problem

Closely related to Chapter 5, the dynamic CRP (DCRP) extends the CRP by considering both stacking and retrieval requests. However, most papers either consider the number of relocations as the objective and/or assume that the schedule of these requests is given and/or restrict the problem to a single row. The first work for the DCRP can be found in [74]. The authors assume that the order of requests is given. They identify the optimal solution to empty a row using an IP similar to the ones proposed for the CRP. Then they suggest four heuristics to select locations for storage requests. Three heuristics are rule-based (and inspired by heuristics developed for the CRP) and one is based on the proposed mathematical formulation. Subsequently, Rei and Pedroso [60] consider a similar problem where items have release and due dates and need to go through a storage area under a given amount of relocations. They show this problem belongs to the complexity class $\mathcal{NP}$ and formulate solutions based on graph-coloring. Motivated by the complexity of the problem, they present a technique to reduce the size of the search space and propose two approaches: the first one is based on multiple simulation methods which use a construction heuristic embedded in a discrete-event simulation model. The second solution proposes a stochastic tree search scheme using best-first-search. Hakan Akyüz and Lee [33] consider the same problem as in [74] where the arrival (departure) sequences of containers to (from) the yard is assumed to be known a priori. A binary IP is developed to solve the DCRP as well as three types of heuristic methods (index based, binary IP based, and beam search heuristics). Borjian et al. [5] introduce a variant of the DCRP by considering a class of flexible service policies to make minor changes in the order of container retrievals (this class is generalized by our flexibilities introduced in Chapter 5).

2.1.4 Other Variants of the CRP

Different objective functions have been considered such as the crane travel time, trucks' waiting times or weighted relocations. López-Plata et al. [50] propose a binary IP to minimize waiting times. Priorities could also be given among groups of

blocks, and Zhu et al. [87] and Tanaka and Takii [66] consider B&B approaches for this case. while de Melo da Silva et al. [15] introduce another variant of the CRP called the Block Retrieval problem. We also mention that Tang et al. [68] solve the CRP using integer programming in the case of steel plates, for which our approach can also be applied. Lee and Lee [47] extend the previous idea and propose a heuristic approach to solve the container retrieval problem. In this problem, all rows in the block are considered and the objective is the crane travel time. The main difference with the YCSP considering only retrieval requests is that the goal of this problem is to retrieve all containers of the block with a pre-defined order that is given initially.

2.2 The Yard Crane Scheduling Problem

The first model for YCSP with a single crane, introduced in [39], considers only retrieval requests and neither storage or relocations enforced by these retrievals. It assumes that containers of the same type are stored in the same row. The retrieval schedule is given by groups of containers and the goal is to minimize crane travel time through the rows (intra rows travel time is not taken into account). Several approaches were tried to solve this problem: Kim and Kim [39] propose the first mixed integer program (MIP). Narasimhan and Palekar [54] show the $\mathcal{NP}$-completeness of the problem, prove some structural properties on the optimal solution and suggest a MIP as well as a branch and bound approach. The best solution in [41] uses encoding and decoding procedures embedded in a neighborhood beam search. Later, Lee et al. [44] consider the same problem for two blocks (one crane per block) and introduce a simulated annealing scheme.

Ng and Mak [55] extend the previous problem by including storage requests but still without considering relocations. They assume that each request has fixed start/end locations and different ready times. In this setting, they assume the processing time of each request and traveling times between requests are given as inputs. Minimizing the sum of request waiting times in this setting is equivalent to a variant of the job shop scheduling problem with inter-job waiting times. They propose a

solution based on a branch-and-bound (B&B) approach. For the same problem, Guo et al. [32] suggest to use A* and RBA* with an admissible heuristic.

Vis and Roodbergen [73] are interested in sequencing of storage and retrieval requests within a block for a single straddle carrier, which they identify as part of planning and scheduling for the transport of unit loads and a generalization of routing an order picker in a warehouse. They assume a special structure for the block: rows are separated by aisles and each row has one I/O point at each end. An important assumption is that the straddle carrier must exit the current row on one of both ends to travel from one row to the other one, which is very restrictive and time consuming. They reformulate the problem as an asymmetric Steiner traveling salesman problem and show that this problem can be solved to optimality by using dynamic programming to link rows together and optimal assignments to solve sub-problems within each row.

The work of Dell et al. [17] is the first to include storage, retrieval and relocation requests as well as the assignment of locations to relocation requests for the YCSP with a single crane. In addition, they schedule housekeeping moves when this is possible. However, they simplify significantly the problem by decomposing the block into areas and considering only the best position within each area for possible storage placement. Moreover, their approach is heuristic-based and considers relocations sequentially. Finally, it is hard to implement in practice as it requires many manual input parameters such as the value of the crane staying idle or the value of placing a container in a stack for each combination of container and stack. For the case of a single crane, their objective is the total crane travel time and they introduce a three-step heuristic. The first step solves a single MIP to prescribe the retrieval requests order and schedule as many storage requests as possible while maximizing idle time for the next steps. Fixing the first step solution, Step 2 solves MIPs sequentially for each relocation and storage move to be done. Finally, step 3 uses a rule based heuristic to schedule housekeeping moves if remaining time is available. They also develop a similar method for two cranes and compares both systems in a simulation study based on these methods.

Gharehgozli et al. [28] consider a setting similar to the one presented in the previous section but disregard unproductive requests (or relocations). In addition, each storage request is associated with a prescribed I/O point (organized in European configuration). They consider a unique crane to carry out storage and retrieval requests while minimizing crane travel time. They prove the $\mathcal{NP}$-hardness of this problem in the case where each retrieval request has a prescribed I/O point. In the rest of their work, they do not make this latter assumption but assume instead that the I/O point for a retrieval request can be uniquely determined when the next request is fixed. Under this assumption, they model the problem as an asymmetric traveling salesman problem and formulate it with a continuous time integer program (IP) with exponentially many constraints. Using specific problem properties, they propose a two-phase solution method to optimally solve the problem: the first phase is a merging algorithm which patch subtours obtained from the assignment relaxation of the problem (relaxing the exponential number of constraints). If the first phase did not find a feasible solution to the original problem, the solution of the first phase is the starting point of a branch-and-bound algorithm. Gharehgozli et al. [29] show that this problem can be solved in polynomial time in the case of two I/O points. The proposed algorithm is based on an improvement of the first phase previously mentioned. However, both papers do not consider relocations moves by saying that the block utilization is low and each container to be retrieved is available on the top of their stacks, which is not realistic in most ports.

Recently, Yuan and Tang [80] solve a similar problem to the one in Chapter 5 but in the setting of coil warehouses. They consider storage and retrieval requests as well as the relocations enforced by retrieval requests. One difference is that the stacking structure enforces triangle blocking constraints instead of typical stack constraints. In addition, some simplification assumptions are made. The I/O points configuration is simpler (European style with one side for storage and one side for retrievals) and they consider $Z = 2$, hence reducing the number of relocations to consider. But most importantly, their objective is minimizing only crane travel time, hence reducing the impact and the complexity of the assignment of relocation and storage locations to

a feasibility problem. In this setting, Yuan and Tang [80] propose a "time-space network flow" MIP formulation where they decompose the scheduling period into stages corresponding to empty/loaded drives of the crane. They also suggest an exact dynamic programming (DP) approach. Both these methods are impractical for large-sized problems (problems with $(X, Y, Z, N) = (3, 5, 2, 12)$ cannot be solved within a 3 minutes' requirement), so they propose an approximate DP method based on the exact DP and value function approximation.

Finally, we mention that recent related works have focused their attention on studying more complex handling systems. Similarly to most works in the single crane setting, these assume that all requests have a given start and end points and the goal is to minimize crane travel time or optimize a combination of other objectives such as truck delays, crane utilization, etc... In this setting, Speer and Fischer [63] give a detailed study of crane cycle times. They review recent papers in twin RMG, double RMG, and triple RMG scheduling. Finally, they compare these systems using a B&B approach and show the impact of considering all parts of the crane cycle times. With respect to assignment of storage locations, Gharehgozli et al. [26] propose a similar approach as in [28] for twin yard cranes where several open locations are considered for each storage request. However, only few open locations are considered for each request and they enforce that each location can be selected by a single storage request, which is not practical. Park et al. [57] consider storage and relocation location assignment using heuristics for twin RMG. However, as in [80], they take into account only the crane travel time to perform the current requests and not future operations. For other related problems such as simulation based models or inter-block crane allocations, we refer the reader to [7, 27].

2.3 Other Optimization Problems in Storage Yards

As we mentioned in Chapter 1, there are two other main problems occurring in storage yards.

Pre-marshalling problem. As we mentioned in Section 1.3, other works have focused on reducing the number of relocations by pre-marshalling, which consists of re-positioning containers to minimizize future relocations during the retrieval process. The first paper in this area by Lee and Hsu [46] introduces the problem for a single row and proposes an integer program using a multi-commodity network flow model as well as a simple heuristic. A neighborhood-based heuristic is developed in [45]. Caserta and Voß [11] formulate the problem in a single row using dynamic programming and apply a local search method called the a corridor method to provide good solutions efficiently. A tree-based heuristic is proposed in [6]. To test methods, Expósito-Izquierdo et al. [19] have developed a generator of instances with different degrees of difficulty. Finally, Tierney and Voß [69] study the robust pre-marshalling problem which also considers uncertainty in the retrieval times of containers.

Storage problem. Even with pre-marshalling and relocating which help minimize the number of relocations, stacking can also have a significant impact in that matter (as shown in Chapter 5). The following papers investigate methods to properly find locations for incoming containers. The first work from [40] consider the storage of export containers in a single row of a block, develop a stochastic dynamic programming model and build upon decision trees. Zhang et al. [85] show the validity of the recursive function of the DP model from [40]. Saurí and Martín [61] develop three new strategies to stack import containers and propose a model to compute the expected number of reshuffles based on container arrival times. Casey and Kozan [12] introduce a mixed-integer programming model to minimize the total amount of time containers spend in the block. Finally, Yu and Qi [79] consider two models; one for unloading import containers, the other one to pre-marshal containers.

Chapter 3

The Container Relocation Problem

3.1 Contributions

This chapter provides two sections dealing with the container relocation problem. In Section 3.2, we define formally the problem.

The major contribution of the first section is a new binary IP formulation for the restricted CRP referred to as CRP-I and presented in Section 3.3. This formulation is novel in several ways. First, CRP-I takes a different modeling approach compared to previous mathematical programming formulations; it builds upon a binary encoding introduced in [9] that has never before been used in exact solution methods. We identify and formulate properties of the encoding as linear equalities or inequalities (Section 3.3.1) in CRP-I and use structural properties of the optimal solution to enhance the tractability of our approach (see Sections 3.3.1 and 3.3.2). The simplicity of our formulation and its adaptability to other related problems could be a key enabler of future advances. Finally, we show through extensive computational experiments on small and medium instances that CRP-I improves upon existing mathematical programming formulations by decreasing significantly the number of variables and constraints. In addition, it outperforms most other exact methods on the instances of [9] (see Section 3.3.3).

In the second section, we study the CRP for large randomly distributed configurations and we show that the ratio between the expected minimum number of

relocations and the number of blocking containers (denoted by $b(.)$) approaches 1. While the problem is known to be $\mathcal{NP}$-hard, this gives strong evidence that the CRP is "easier" to solve for large instances, and that heuristics can find near-optimal solutions on average. Average case analysis of CRP is fairly new. The only other paper found in the literature is by Olsen and Gross [56]. They also provide a probabilistic analysis of the asymptotic CRP when both the number of stacks and tiers grow to infinity. They show that there exists a polynomial time algorithm that solves this problem close to optimality with high probability. Our model departs from theirs in two aspects: (i) We keep the maximum number of tiers a constant whereas in [56] it also grows. Our assumption is motivated by the fact that the maximum tier is limited by the crane height, and it cannot grow arbitrarily; and (ii) We assume the ratio of the number of containers initially in the configuration to the configuration size (i.e., number of stacks) stays constant (i.e., the configuration is almost full at the beginning) and is equal to h. On the other hand, in [56], the ratio of the number of containers initially in the configuration to the configuration size decreases (and it approaches zero) as the number of stacks grows. In other words, in their model, in large configurations, the configuration is underutilized, thus simpler to solve.

3.2 Problem Description

The container relocation problem (CRP) usually models one row using a two-dimensional array of size (T, S), where S is the number of stacks, and T is the maximum tier, i.e., the maximum number of containers in a stack as limited by the height of the crane. Each element of this array represents a potential slot for a container, and the slot contains a number only if a container is currently stored in this slot. Stacks are numbered from left (1) to right (S) and tiers from bottom (1) to top (T). We refer to this array as a *configuration*. The common assumptions of the CRP are the following:

> $\mathbf{A_1}$: The initial configuration has T tiers, S stacks, and C containers. In order for the problem to always be feasible, we suppose that the triplet (T, S, C) satisfies $0 \leqslant C \leqslant ST - (T - 1)$. Indeed, $T - 1$ empty slots would be needed

to relocate a maximum of $T-1$ blocking containers above the target container (see [74, 10]).

$\mathbf{A_2}$: A container can be retrieved/relocated only if it is in the topmost tier of its stack, i.e., no other container is blocking it.

$\mathbf{A_3}$: A container can be relocated only if it is blocking the target container. Container c is said to be *blocking* if there exists container d in a lower tier of the same stack such that $d < c$. This assumption was suggested in [10], and the problem under this assumption is commonly referred to as the *restricted* CRP. Most studies focus on this restricted version, because it is the current practice in many yards, and it helps decrease the dimensionality of the problem, while not losing much optimality (see [58]). As is common practice, we will not mention the term "restricted" in the rest of this chapter even though we always assume A_3.

$\mathbf{A_4}$: The cost of relocating a container from a stack does not depend on the stack to which the container is relocated. It motivates the objective of minimizing the number of relocations, since the cost of each relocation can be normalized to 1. In addition, this allows us in the next chapter to consider the stacks of a configuration as interchangeable. Note that this assumption is not required for most results in Chapters 3 and 4, hence our approaches could be easily extended to the case when Assumption A_4 does not hold.

$\mathbf{A_5}$: The retrieval order of containers is known, so that each container can be labeled from 1 to C, representing the departure order, i.e., Container 1 is the first one to be retrieved, and C the last one.

The CRP involves finding a sequence of moves to retrieve Containers $1, 2, \ldots, C$ (respecting the order) with a minimum number of relocations. Figure 3-1 provides a simple example of the CRP.

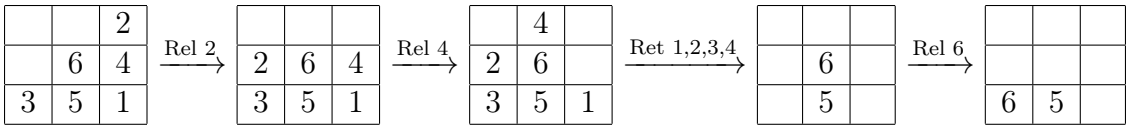


Figure 3-1: Configuration for the CRP with 3 tiers, 3 stacks, and 6 containers. The optimal solution performs 3 relocations: relocate the container labeled 2 from Stack 3 to Stack 1 on the top of the container labeled 3; relocate 4 from 3 to 2 on the top of 6; retrieve 1; retrieve 2; retrieve 3; retrieve 4; relocate 6 from 2 to the empty Stack 1; retrieve 5; finally, retrieve 6.

3.3 A New Binary Formulation Based on a Binary Encoding of Configurations

This section is organized as follows. Section 3.3.1 introduces some preliminary concepts of the binary encoding and structural properties of the optimal solution. Section 3.3.2 presents formulation CRP-I and suggests improvements for this new formulation, such as a revisited preprocessing step. Section 3.3.3 presents the results of computational experiments.

3.3.1 Preliminaries

Binary encoding

Most integer program formulations mentioned in Section 2.1 use the matrix representation of a configuration, i.e., they introduce binary variables of the type y_{tscn} which indicates if container c is in tier t of stack s before the n^{th} move is performed.

Our binary IP models a configuration using an enhanced version of the binary encoding introduced in [9]. First, Caserta et al. [9] define S *artificial* containers identified by integers from $C+1, \ldots, C+S$, where artificial container c is under all containers in stack $c-C$ (see Figure 3-2). These artificial containers are used to keep track of which containers are in which stacks. Using these containers, they encode the classical matrix representation of the configuration into a binary matrix denoted by A of size $(C+S) \times (C+S)$. We consider the same encoding but decrease its size to $(C+S) \times C$. Rows indexed from $1, \ldots, C$ and columns of A represent the C *real*

containers, while rows indexed from $C+1, \ldots, C+S$ correspond to the S artificial containers. Formally, the binary encoding $A \in \{0,1\}^{(C+S)\times C}$ is defined such that such that

$$\forall\ c \in \{1, \ldots, C+S\}, d \in \{1, \ldots, C\},\ A_{c,d} = \begin{cases} 1 & \text{if container } c \text{ is below container } d, \\ 0 & \text{otherwise.} \end{cases}$$

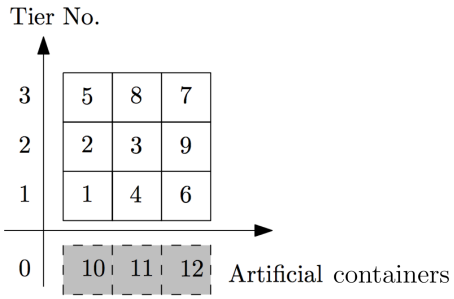


Figure 3-2: Configuration including artificial containers (example from [9]).

The binary encoding of the configuration in Figure 3-2 is

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

By using the matrix A in this example, we can quickly access valuable information.

For instance, consider container 1. The first row indicates that container 1 is below containers 2 and 5 and the first column shows that container 1 is above artificial container 10. Because this container represents stack $10 - 9 = 1$, it indicates that container 1 is in the first stack.

Logical constraints of the CRP. We now describe the main logical constraints of any feasible configuration the CRP. For each of these constraints, we use the aforementioned binary encoding to formulate it as a linear equality or inequality to be embedded into our mathematical programming formulation. Given a target container $n \in \{1, \ldots, C\}$, any feasible configuration for the CRP respects the following logical constraints:

1. **Only containers $n, \ldots, C$ remain in the configuration.** Thus, we only need to keep track of rows $c \in \{n, \ldots, C+S\}$ and columns $d \in \{n, \ldots, C\}$ since:

$$\forall\ c \in \{1, \ldots, n-1\},\ \begin{cases} \forall\ d \in \{1, \ldots, C\},\ A_{c,d} = 0, \\ \forall\ d \in \{1, \ldots, C+S\},\ A_{d,c} = 0. \end{cases}$$

   Therefore, we consider that $A \in \{0,1\}^{(C+S-n+1)\times(C-n+1)}$.

2. **A container is in one stack**, and so is blocking exactly one artificial container:

$$\forall\ c \in \{n, \ldots, C\},\ \sum_{s=1}^{S} A_{C+s,c} = 1.$$

3. **A container cannot be blocking itself**, so the upper diagonal of A only contains zeros:

$$\forall\ c \in \{n, \ldots, C\},\ A_{c,c} = 0.$$

4. **Among two distinct containers c and d, only one can be below the**

other one:

$$\forall c, d \in \{n, \ldots, C\},\ c \neq d,\ A_{c,d} + A_{d,c} \leqslant 1.$$

5. **If two distinct containers c and d are in the same stack s, then either c is below d or d is below c** ($A_{c,d} + A_{d,c} \geqslant 1$). We formulate this as:

$$\forall\ s \in \{1, \ldots, S\},\ c, d \in \{n, \ldots, C\},\ c \neq d,\ A_{c,d} + A_{d,c} \geqslant A_{C+s,c} + A_{C+s,d} - 1.$$

There are three cases to consider:

- If c and d are in stack s, i.e., $A_{C+s,c} = A_{C+s,d} = 1$, then this equation implies $A_{c,d} + A_{d,c} \geqslant 1$. Using the previous logical constraint, we have $A_{c,d} + A_{d,c} = 1$, which is the relation we are looking for.
- If either c or d is not in stack s, then this equation implies $A_{c,d} + A_{d,c} \geqslant 0$, which holds in any case since $A_{c,d}, A_{d,c} \in \{0, 1\}$.
- If c and d are not in stack s, then this equation implies $A_{c,d} + A_{d,c} \geqslant -1$, which holds in any case since $A_{c,d}, A_{d,c} \in \{0, 1\}$.

6. **If two distinct containers c and d are in different stacks then neither c is below d or d below c** ($A_{c,d} = 0$ and $A_{d,c} = 0$). This constraint can be written as:

$$\forall\ s \in \{1, \ldots, S\},\ c, d \in \{n, \ldots, C\},\ c \neq d,\ A_{c,d} + A_{d,c} \leqslant 2 - A_{C+s,c} - \sum_{\substack{r=1 \\ r \neq s}}^{S} A_{C+r,d}.$$

Similarly to the previous logical constraint, we consider three cases:

- If c is in stack s but not d, i.e., $A_{C+s,c} = 1$ and $\sum_{\substack{r=1 \\ r \neq s}}^{S} A_{C+r,d} = 0$, then this equation implies $A_{c,d} + A_{d,c} \leqslant 0$. Since $A_{c,d}, A_{d,c} \in \{0, 1\}$, we must have $A_{c,d} = 0$ and $A_{d,c} = 0$ which is the constraint we are looking for.

- If both or neither c and d are in stack s, then this equation implies $A_{c,d} + A_{d,c} \leqslant 1$ (which holds since it is the expression of the fourth logical constraint of the CRP expressed with the binary encoding).
- If c is not in stack s but d is, then this equation implies $A_{c,d} + A_{d,c} \leqslant 2$ which always holds since $A_{c,d}, A_{d,c} \in \{0, 1\}$.

7. **The number of relocations needed to retrieve the target container is the number of containers directly blocking it.** Since the number of containers blocking c is given by $\sum_{d=n+1}^{C} A_{c,d}$. In particular, the number of relocations to retrieve the target container n is $\sum_{d=n+1}^{C} A_{n,d}$.

8. **The number of containers in each stack is at most $\boldsymbol{T}$**. The height of stack $s \in \{1, \ldots, S\}$ is the number of containers blocking artificial container $C + s$, i.e., $\sum_{d=n+1}^{C} A_{C+s,d}$, so

   $$\forall\ s \in \{1, \ldots, S\},\ \sum_{d=n+1}^{C} A_{C+s,d} \leqslant T.$$

9. **The total number of blocking containers in $\boldsymbol{A}$, denoted by $\boldsymbol{b}(\boldsymbol{A})$, is given by $\boldsymbol{b}(\boldsymbol{A}) = \sum_{\boldsymbol{d=n+1}}^{\boldsymbol{C}} \boldsymbol{B_d}$, where $\boldsymbol{B_d} \in \{\boldsymbol{0, 1}\}$ is a binary indicator that container $\boldsymbol{d}$ is blocking**, which can be computed using the binary encoding simply as follows:

   $$\forall\ d \in \{n+1, \ldots, C\},\ c \in \{n, \ldots, d-1\},\ B_d \geqslant A_{c,d}.$$

   Note that the total number of blocking containers is not $\sum_{c=n}^{C-1} \sum_{d=c+1}^{C} A_{c,d}$, as a blocking container can be blocking several containers but should count only once.

Apart from offering a simple representation of a given configuration, this binary encoding presents another great advantage. Tracking the evolution of the configuration with the classical matrix representation is fairly intuitive but requires a significant number of constraints and seems to be one of the bottlenecks of previous formulations. Our representation allows us to express efficiently the transformation of the matrix A when performing the necessary relocations to retrieve the target container.

Let $n \in \{1, \dots, C-1\}$, we denote $A \in \{0,1\}^{(C+S-n+1)\times(C-n+1)}$ (respectively, $A' \in \{0,1\}^{(C+S-n)\times(C-n)}$) the binary encoding of the configuration when n is the target container (respectively, when $n+1$ is the target container and relocations to retrieve n have been performed). The logical constraints of the CRP enforce that A' and A should verify the three following rules:

A. **The restricted assumption states that if container c is not blocking the target container n then c cannot be relocated.** Therefore all containers below c cannot be relocated either, hence columns c in A and A' should be identical, i.e., if $A_{n,c} = 0$, then $\forall\ d \in \{n+1, \dots, C\}$, $A'_{d,c} = A_{d,c}$. This can be formulated by the following two inequalities:

$$\forall\ c, d \in \{n+1, \dots, C\},\ \begin{cases} A'_{d,c} \leqslant A_{d,c} + A_{n,c}, \\ A'_{d,c} \geqslant A_{d,c} - A_{n,c}. \end{cases}$$

If $A_{n,c} = 0$, both inequalities imply the constraint we are looking for. If $A_{n,c} = 1$, the inequalities imply $A_{d,c} - 1 \leqslant A'_{d,c} \leqslant A_{d,c} + 1$ which always holds since $A_{d,c} - 1 \leqslant 0$, $A_{d,c} + 1 \geqslant 1$ and $A'_{d,c} \in \{0,1\}$.

B. **Any relocated container cannot stay in the same stack,** which can be formulated as:

$$\forall\ c \in \{n+1, \dots, C\},\ s \in \{1, \dots, S\},\ A_{n,c} + A_{C+s,c} + A'_{C+s,c} \leqslant 2.$$

We consider two cases to justify this formulation:

- If container c is blocking the target container n (hence relocated) and is in

stack s, i.e., $A_{n,c} = A_{C+s,c} = 1$, then the inequality implies that $A'_{C+s,c} = 0$ which is the logical constraint we are aiming for.

- If either container c is not blocking the target container n or is not in stack s, then the inequality implies that $A'_{C+s,c} \leqslant 1$ which always holds.

C. **If two distinct containers c and d are blocking container n (hence relocated) and d is above c when n is the target container, then d cannot remain above c (due to the LIFO policy of stacks)**, i.e., if $A_{n,c} = A_{n,d} = A_{c,d} = 1$, then $A'_{c,d} = 0$. This can be expressed as:

$$\forall\ c, d \in \{n+1, \ldots, C\},\ c \neq d,\ A_{n,c} + A_{n,d} + A_{c,d} + A'_{c,d} \leqslant 3.$$

Again, we justify this formulation by considering two cases:

- if $A_{n,c} = A_{n,d} = A_{c,d} = 1$, the inequality clearly implies $A'_{c,d} = 0$.
- if either $A_{n,c} = 0$, $A_{n,d} = 0$ or $A_{c,d} = 0$, then the inequality implies that $A'_{c,d} \leqslant 1$ which always holds.

Minimum number of relocations when $C \leqslant S$

We use a result from [24] (see Lemma 5) which we restate below with the notations of this section.

Lemma 1 ([24])**.** *Consider a configuration with T tiers, S stacks, and C containers with binary representation $A \in \{0,1\}^{(C+S)\times C}$. Let $b(A)$ be the number of blocking containers in A which can be computed as*

$$b(A) = \sum_{d=2}^{C} B_d \text{ with } B_d \in \{0,1\}\ ,\ B_d \geqslant A_{c,d}, \forall\ c \in \{1, \ldots, d-1\}.$$

If $C \leqslant S$, then the minimum number of relocations to retrieve all containers from A is equal to $b(A)$.

Lemma 1 states that when the number of containers is fewer than the number of stacks ($C \leqslant S$), the minimum total number of relocations is simply the number of

blocking containers. From now on, we suppose that $C > S$ and we define

$N = C - S + 1 > 1.$

Let us explain the use of Lemma 1. Let $A^{(1)}$ (respectively, $A^{(N)}$) be the binary encoding of an initial configuration (respectively, the configuration after $N-1$ retrievals). We define $r\left(A^{(1)}, A^{(N)}\right)$ to be the minimum number of relocations to reach $A^{(N)}$ from $A^{(1)}$. By definition $A^{(N)} \in \{0,1\}^{(C'+S)\times C'}$, where $C' = C - N + 1 = S \leqslant S$. Hence we can apply Lemma 1 to $A^{(N)}$ and the minimum number of relocations to retrieve all containers from $A^{(N)}$ is equal to $b\left(A^{(N)}\right)$. Therefore, the minimum number of relocations to retrieve all containers from $A^{(1)}$ is $r\left(A^{(1)}, A^{(N)}\right) + b\left(A^{(N)}\right)$. Usually, methods track all relocations until there are no containers left in the configuration. Lemma 1 allows us to restrict our focus on the first $N-1$ retrievals as the problem becomes "trivial" when there are only S containers left.

3.3.2 New Binary IP Formulation

Using Section 3.3.1, we can now present our novel formulation that we denote by CRP-I. We suppose that we are given as inputs a feasible triplet (T, S, C) and $A^{(1)}$ the binary encoding of the initial configuration (which verify all the properties of the binary encoding). CRP-I minimizes the total number of relocations to retrieve all containers from a given initial configuration while enforcing all the previously mentioned logical constraints of the CRP. To do so, CRP-I uses two kinds of variables,

$\forall\, n \in \{1,\ldots,N\}, c \in \{n,\ldots,C+S\}, d \in \{n,\ldots,C\},$

$$a_{n,c,d} = \begin{cases} 1 & \text{if container } c \text{ is below container } d \text{ when } n \text{ is the target container,} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\forall\, d \in \{N+1, \ldots, C\},$$

$$b_d = \begin{cases} 1 & \text{if container } d \text{ is blocking after } N-1 \text{ retrievals,} \\ 0 & \text{otherwise.} \end{cases}$$

Binary variables a are used to keep track of the binary encoding of the configurations and compute the number of relocations incurred for the first $N-1$ retrievals. Note that we use the first logical constraint of the CRP expressed with the binary encoding to limit the number of variables as n increases. Binary variables b are used to compute the number of blocking containers in the configuration after the $N-1$ retrievals. The number of variables is of the order of C^3, where the exact number is

$$S - 1 + \sum_{n=1}^{N-1} (C + S - n + 1) \times (C - n + 1) = \frac{1}{3}C^3 + \frac{1+S}{2}C^2 + \frac{1+3S}{6}C - \frac{6 - 5S - 6S^2 + 5S^3}{6}.$$

The objective and constraints of CRP-I are presented below

$$Min \left(\sum_{n=1}^{N-1} \sum_{d=n+1}^{C} a_{n,n,d} + \sum_{d=N+1}^{C} b_d \right)$$

s.t.

$$a_{1,c,d} = A^{(1)}_{c,d}, \qquad \forall\, c \in \{1, \ldots, C+S\}, d \in \{1, \ldots, C\} \tag{3.1}$$

$$\sum_{s=1}^{S} a_{n,C+s,c} = 1, \qquad \forall\, n \in \{2, \ldots, N\}, c \in \{n, \ldots, C\} \tag{3.2}$$

$$a_{n,c,c} = 0, \qquad \forall\, n \in \{2, \ldots, N\}, c \in \{n, \ldots, C\} \tag{3.3}$$

$$a_{n,c,d} + a_{n,d,c} \leqslant 1, \qquad \forall\, n \in \{2, \ldots, N\}, c \in \{n, \ldots, C\}, d \in \{n, \ldots, C\} \setminus \{c\} \tag{3.4}$$

$$a_{n,c,d} + a_{n,d,c} \geqslant a_{n,C+s,c} + a_{n,C+s,d} - 1, \\ \forall\ n \in \{2,\dots,N\}, s \in \{1,\dots,S\}, c \in \{n,\dots,C\}, d \in \{n,\dots,C\} \setminus \{c\} \tag{3.5}$$

$$a_{n,c,d} + a_{n,d,c} \leqslant 2 - a_{n,C+s,c} - \sum_{\substack{r=1 \\ r \neq s}}^{S} a_{n,C+r,d}, \\ \forall\ n \in \{2,\dots,N\}, s \in \{1,\dots,S\}, c \in \{n,\dots,C\}, d \in \{n,\dots,C\} \setminus \{c\} \tag{3.6}$$

$$\sum_{d=n}^{C} a_{n,C+s,d} \leqslant T, \qquad \forall\ n \in \{2,\dots,N\}, s \in \{1,\dots,S\} \tag{3.7}$$

$$b_d \geqslant a_{N,c,d}, \qquad \forall\ d \in \{N+1,\dots,C\}, c \in \{N,\dots,d-1\} \tag{3.8}$$

$$a_{n+1,d,c} \leqslant a_{n,d,c} + a_{n,n,c}, \\ \forall\ n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, d \in \{n+1,\dots,C+S\} \setminus \{c\} \tag{3.9}$$

$$a_{n+1,d,c} \geqslant a_{n,d,c} - a_{n,n,c}, \\ \forall\ n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, d \in \{n+1,\dots,C+S\} \setminus \{c\} \tag{3.10}$$

$$a_{n,n,c} + a_{n,C+s,c} + a_{n+1,C+s,c} \leqslant 2, \\ \forall\ n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, s \in \{1,\dots,S\} \tag{3.11}$$

$$a_{n,n,c} + a_{n,n,d} + a_{n,c,d} + a_{n+1,c,d} \leqslant 3, \\ \forall\ n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, d \in \{n+1,\dots,C\} \setminus \{c\}. \tag{3.12}$$

$$a_{n,c,d} \in \{0,1\}, \qquad \forall\, n \in \{1,\ldots,N\}, c \in \{n,\ldots,C+S\}, d \in \{n,\ldots,C\} \tag{3.13}$$

$$b_d \in \{0,1\}, \qquad \forall\, d \in \{N+1,\ldots,C\} \tag{3.14}$$

The objective function of CRP-I minimizes the total number of relocations. As suggested in Section 3.3.1, the objective function has two main components. The first term counts the number of relocations incurred for the first $N-1$ retrievals, i.e., $r\left(A^{(1)}, A^{(N)}\right)$. When the target container is $n \in \{1,\ldots,N-1\}$, the seventh logical constraint of the CRP states how to compute the number of containers blocking the target container n using the binary encoding, hence the number of relocations to retrieve n. The second term counts the number of blocking containers in the configuration after the first $N-1$ retrievals, i.e., $b\left(A^{(N)}\right)$, as suggested by Lemma 1.

Constraint (3.1) initializes the variable a for $n = 1$ with the input binary encoding $A^{(1)}$. Constraints (3.2), (3.3), (3.4), (3.5), (3.6) and (3.7) correspond to the second, third, fourth, fifth, sixth and eighth logical constraints of the CRP formulated using the binary encoding. Constraint (3.8) enforces the relation between variables b and a, which is explained in the ninth logical constraint of the CRP. The four next constraints enforce the rules to update the binary encoding between retrievals. Constraints (3.9) and (3.10) correspond to rule A, Constraint (3.11) corresponds to rule B and Constraint (3.12) corresponds to rule C. Finally, constraints (3.13) and (3.14) ensure that variables are binary variables.

We point out that variables of CRP-I only keep track of the configuration states through the binary encoding, and it could seem that no decisions are made. Actually, decisions are implicitly taken when a state of configuration is chosen after each retrieval and the cost of these decisions (i.e., the number of relocations) can easily be induced. Let $n \in \{1,\ldots,N\}$ and $c \in \{n,\ldots,C\}$, then we define:

$$s_n(c) = \sum_{s=1}^{S} s a_{n,C+s,c} \qquad \textit{and} \qquad t_n(c) = \sum_{d=n}^{C+S} a_{n,d,c}, \tag{3.15}$$

such that $s_n(c)$ (respectively $t_n(c)$) is the stack (respectively tier) at which container c is when container n is the target container. Using these notations, any feasible a for CRP-I defines a feasible sequence of configurations starting with $A^{(1)}$ and ending with a configuration with S containers. Conversely, we have derived the constraints of CRP-I from logical constraints of the CRP, then the binary encoding of a feasible sequence of configurations starting with $A^{(1)}$ and ending with a configuration with S containers is feasible for CRP-I.

Finally, we believe that this modeling approach can be easily generalized to different objective functions. We provide three extensions in Appendix A.1.

Minor improvements

We suggest here some features that we use together with CRP-I when calling commercial solvers. First, since the optimal objective value is an integer, we can set *the absolute gap of the optimization solver to* $1 - \epsilon$ *for* $\epsilon > 0$. Indeed, if we can guarantee that the best feasible solution found so far with objective z_{best} is such that $z_{best} - z_{opt} \leqslant 1 - \epsilon$, then by integrality of z_{best} and z_{opt}, this ensures that $z_{best} = z_{opt}$. Second, Caserta et al. [10] suggest a Minmax heuristic which is commonly used as a good upper bound. *We include this solution as a first incumbent to the solver.*

The flexibility of CRP-I allows us to take into account some existing improvements suggested in the literature to enhance the efficiency of integer programs for the CRP. For instance, *we can simply adapt one of the preprocessing steps suggested by Zehendner et al. [81].* For each container $c \in \{1, \ldots, C\}$, one can easily compute the first target container $\pi_c \in \{1, \ldots, c\}$ for which c is moved. Therefore, all containers below c, including c, stay in their initial position until container π_c is the target container. Thus the c^{th} column of the binary encoding does not change until π_c is the current target container, i.e.,

$$a_{n,d,c} = A^{(1)}_{d,c} \qquad \forall\, n \in \{1, \ldots, \min\{\pi_c - 1, N\}\}, c \in \{n, \ldots, C\}, d \in \{n, \ldots, C + S\}.$$

Although we do not use it in our experiments, we mention here that we can also easily

adapt constraint (B) *from [81]* as

$$\sum_{d=n+1}^{C} a_{n,n,d} \leqslant UB_n, \qquad \forall\, n \in \{1, \ldots, N-1\},$$

where the computation of UB_n is detailed in [81].

3.3.3 Computational Experiments

As mentioned in Section 3.1, we test our binary model on the instances introduced in [9]. These instances are defined by two values (T', S), where S is the number of stacks and T' the initial number of containers per stack (identical for all stacks). It is assumed that the maximum number of tiers T is taken to be $T = T' + 2$. Each class associated with a pair (T', S) contains 40 instances, each representing different initial configurations with $C = S \times T'$ containers.

Experiments are carried out on a Dell C6300 with an Intel E5-2690 v4 2.6 GHz processor, 4 CPUs, 8.00 GB of RAM and the programming language is Julia 0.5.0. We use Gurobi 7.0.1. to solve the binary programming model CRP-I enhanced by the gap (with $\epsilon = 0.01$), and the incumbent and preprocessing improvements described in the previous section. We set a time limit of *3600 seconds per instance* for the solver. We also define the memory limit to be when *the product of the number of variables with the number of constraints is greater than* 8×10^{10}. This definition is different than in [81], but it has the merit of not depending on the computer system, allowing for an easier comparison with future studies. All instances and scripts are available at https://github.com/vgalle/binaryIP_CRP. We present our results as in [81] to ease the comparison with this state-of-the art integer programming solution. Finally, we also compare our results with the integer program and the branch and bound algorithms presented in [18, 20, 42, 66] on the first 5 instances of several classes, as these are the only instances for which they provide results in their papers.

(T', S)	C	Avg. LB	Avg. UB	Avg. gap	Nb. trivial
$(3, 3)$	9	4.725	5.075	0.35	27
$(3, 4)$	12	5.85	6.30	0.45	26
$(3, 5)$	15	6.75	7.05	0.30	28
$(3, 6)$	18	8.275	8.45	0.175	36
$(3, 7)$	21	9.10	9.325	0.225	33
$(3, 8)$	24	10.45	10.725	0.275	32
$(4, 4)$	16	9.40	10.975	1.575	8
$(4, 5)$	20	12.15	13.55	1.40	11
$(4, 6)$	24	13.225	14.675	1.45	11
$(4, 7)$	28	15.20	16.90	1.70	9
$(5, 4)$	20	13.575	16.75	3.175	4
$(5, 5)$	25	17.00	21.225	4.225	1
$(5, 6)$	30	20.05	24.25	4.20	2
$(5, 7)$	35	22.425	26.325	3.90	5
$(5, 8)$	40	25.60	29.60	4.00	4
$(5, 9)$	45	28.525	32.35	3.825	3
$(5, 10)$	50	31.35	35.50	4.15	2
$(6, 6)$	36	26.95	35.90	8.95	0
$(6, 10)$	60	41.525	49.85	8.325	2
$(10, 6)$	60	56.60	101.25	44.65	0
$(10, 10)$	100	84.125	139.275	55.15	0

Table 3.1: Difficulty of instances from [9].

Difficulty of instances

Table 3.1 shows several key values indicating the difficulty of the instances. It displays the problem class (T', S), the number of containers in each class C, the average lower bound from [87] (Avg. LB), the average upper bound from [10] (Avg. UB) and the average gap between these two values (Avg. gap). Most importantly, it gives how many instances can be considered as trivial (Nb. trivial), i.e., when the upper bound is equal to the lower bound, hence the integer program is not required to solve these instances. We mention that Table 3.1 confirms exactly the results from [81]. We mainly provide this table for the sake of clarity associated with the next results.

Results of CRP-I and comparison with BRP-II-A

We provide the results of CRP-I aggregated by class of instances in Table 3.2. This table recalls the number of trivial instances per class (Nb. trivial) and displays the number of instances that were solved to optimality (Nb. solved non-trivial), the average and standard deviation of CPU times in seconds (Avg. CPU time and Std. CPU time). Finally, it differentiates instances not solved optimally either due to time or memory limit. We provide in parenthesis the results from [81] to make a comparison both in terms of the number of instances solved optimally as well as CPU times.

(T', S)	Nb. trivial	Nb. solved non-trivial	Avg. CPU time (sec.)	Std. CPU time (sec.)	Nb. time limit	Nb. memory limit
$(3, 3)$	27	13 (13)	0.1 (0.1)	0.1 (0.0)	0 (0)	0 (0)
$(3, 4)$	26	14 (14)	0.1 (0.3)	0.1 (0.3)	0 (0)	0 (0)
$(3, 5)$	28	12 (12)	0.1 (0.8)	0.1 (0.7)	0 (0)	0 (0)
$(3, 6)$	36	4 (4)	0.4 (4.2)	0.2 (3.0)	0 (0)	0 (0)
$(3, 7)$	33	7 (7)	0.6 (5.8)	0.4 (3.4)	0 (0)	0 (0)
$(3, 8)$	32	8 (8)	1.3 (11.2)	0.7 (6.0)	0 (0)	0 (0)
$(4, 4)$	8	32 (32)	0.1 (1.2)	0.1 (0.7)	0 (0)	0 (0)
$(4, 5)$	11	29 (29)	0.5 (5.8)	0.3 (3.7)	0 (0)	0 (0)
$(4, 6)$	11	29 (29)	2.3 (16.1)	5.4 (28.3)	0 (0)	0 (0)
$(4, 7)$	9	31 (31)	6.4 (90.1)	12 (197.8)	0 (0)	0 (0)
$(5, 4)$	4	36 (36)	1.8 (19.9)	4.5 (38.3)	0 (0)	0 (0)
$(5, 5)$	1	**39** (38)	41.9 (369.3)	177.6 (798.9)	0 (1)	0 (0)
$(5, 6)$	2	**37** (31)	68.0 (524.3)	166.2 (719.4)	1 (6)	0 (0)
$(5, 7)$	5	**31** (19)	170.9 (487.7)	367.4 (609.8)	4 (2)	0 (14)
$(5, 8)$	4	**29** (5)	590.2 (749.4)	820.6 (535.0)	7 (0)	0 (31)
$(5, 9)$	3	**23** (2)	658.2 (126.1)	799.1 (79.5)	14 (0)	0 (35)
$(5, 10)$	2	**20** (0)	1111.0 (n.a.)	1056.7 (n.a.)	18 (0)	0 (38)
$(6, 6)$	0	**19** (7)	441.7 (1466.5)	723.8 (1092.9)	21 (10)	0 (23)
$(6, 10)$	2	0 (0)	n.a. (n.a.)	n.a. (n.a.)	0 (0)	38 (38)
$(10, 6)$	0	0 (0)	n.a. (n.a.)	n.a. (n.a.)	0 (0)	40 (40)
$(10, 10)$	0	0 (0)	n.a. (n.a.)	n.a. (n.a.)	0 (0)	40 (40)

Table 3.2: Computational results of CRP-I on non-trivial instances (in parenthesis the results from [81]). In bold, the classes for which CRP-I solves more instances optimally than BRP-II-A.

Our experiment shows that we can now solve almost all instances with sizes lower than $(5, 7)$ within minutes on average. We can also solve a majority of instances for classes $(5, 8$ to $10)$ and some in $(6, 6)$. However, all the other classes reach the memory limit threshold. In terms of number of instances solved optimally, our method clearly outperforms BRP-II-A, specially for the classes $(5, 7$ to $10)$ and $(6, 6)$. The average

CPU time and its variability are also dramatically improved by CRP-I compared to BRP-II-A. Note that there are much fewer instances solved by BRP-II-A than CRP-I for classes $(5, 8)$ and $(5, 9)$, hence explaining the lower variability of CPU time of BRP-II-A for these classes.

(T', S)	Nb. binary variables	Avg. nb. constraints	Avg. node iterations	Difference of LP relax. nb. blocking	LB
$(3, 3)$	408 (546.8)	2,326.4	0	1.10	-0.14
$(3, 4)$	927 (2,140.1)	6,730.7	0	0.86	-0.42
$(3, 5)$	1,764 (5,236.0)	15,389.8	0	0.74	-0.68
$(3, 6)$	2,995 (16,170.8)	30,303.2	0	0.70	-0.80
$(3, 7)$	4,696 (25,576.0)	54,364.3	0	0.79	-0.93
$(3, 8)$	6,943 (37,248.0)	90,340.4	0	0.50	-0.63
$(4, 4)$	2,005 (5,755.5)	16,383.5	0	1.36	-0.83
$(4, 5)$	3,844 (19,380.2)	37,413.0	0	1.06	-0.94
$(4, 6)$	6,560 (32,718.8)	74,130.1	3.5	1.03	-1.32
$(4, 7)$	10,324 (72,055.6)	132,182.1	0	0.82	-1.06
$(5, 4)$	3,675 (15,161.6)	32,361.9	22.8	1.87	-1.00
$(5, 5)$	7,074 (42,530.7)	73,823.8	420.7	2.09	-1.70
$(5, 6)$	12,105 (112,706.7)	145421.6	77.5	1.67	-1.70
$(5, 7)$	19,088 (198,902.1)	260,202.2	25.6	1.72	-2.08
$(5, 8)$	28,343 (352,122.7)	431,277.6	18.2	1.73	-2.34
$(5, 9)$	40,190 (583,790.6)	675,140.5	10.7	1.95	-3.65
$(5, 10)$	54,949 (884,834.6)	1,009,631.0	12.2	2.19	-4.86
$(6, 6)$	20,062 (243,620.9)	253,368.7	466.4	5.62	-5.96
$(6, 10)$	91,384 (2,058,673.2)	1,750,000.0	n.a.	n.a.	n.a.
$(10, 6)$	84,650 (-)	1,170,272.8	n.a.	n.a.	n.a.
$(10, 10)$	388,124 (-)	8,090,000.0	n.a.	n.a.	n.a.

Table 3.3: Efficiency indicators of CRP-I on non-trivial instances (in paranthesis the number of variables from [81] with preprocessing).

Table 3.3 presents alternative indicators which only depend on the binary formulation properties and not on the computer features. Compared to those in Table 3.2, these indicators provide a more unbiased estimation of the performance of CRP-I. We provide the number of binary variables (Nb. binary variables), the average number of constraints (Avg. nb. constraints) over all 40 instances of each class. Note that the number of binary variables is constant in a given class of instances. In addition, we show the average number of node iterations done by the solver (Avg. node itera-

tions) for instances solved optimally. We can observe that the number of variables is very reasonable for all classes (expect $(10, 10)$) and that CRP-I decreases this number substantially compared to BRP-II-A (by a factor varying from 1.34 to 22), even with their proposed preprocessing step. This can partly explain the performances reported in Table 3.2. However, the bottleneck of our model seems to be the substantial growth of the number of constraints as instance size grows. Future work could be to study a row-generation approach for the CRP-I model. Finally, the last two columns of Table 3.3 provide the difference between the LP relaxation objective function and two other lower bounds (the number of blocking containers and the lower bound from [87]). We notice that the LP relaxation is on average always between these two lower bounds, which leads to say that the LP relaxation is quite good but may be further improved by adding efficient cuts which could be using lower bounds properties.

Efficiency of the upper bound and hardness of the CRP

(T', S)	Nb. solved non-trivial	Avg. Opt.	Avg. Gap UB-Opt.	Nb. UB optimal
$(3, 3)$	13	6.77	0.23	10
$(3, 4)$	14	7.5	0.36	10
$(3, 5)$	12	8.75	0.08	11
$(3, 6)$	4	11.5	0.50	2
$(3, 7)$	7	11.71	0.29	5
$(3, 8)$	8	12.25	0.38	5
$(4, 4)$	32	10.69	0.97	12
$(4, 5)$	29	13.59	0.83	14
$(4, 6)$	29	14.55	0.90	15
$(4, 7)$	31	16.9	1.00	12
$(5, 4)$	36	15.61	1.47	9
$(5, 5)$	39	18.79	2.44	7
$(5, 6)$	37	22.24	2.22	5
$(5, 7)$	31	24.23	2.13	4
$(5, 8)$	29	27.48	1.86	9
$(5, 9)$	23	29.52	2.09	5
$(5, 10)$	20	32.00	1.8	5
$(6, 6)$	19	28.05	3.58	1

Table 3.4: Efficiency of the upper bound and hardness of the CRP on non-trivial instances.

Table 3.4 provides the number of non-trivial instances solved optimally by CRP-I and for these instances,the average optimal number of relocations (Avg. Opt.), the average gap between the upper bound and the optimal solution (Avg. Gap UB-Opt) and the number of instances for which the upper bound is optimal (Nb. UB optimal). These key factors can help estimate the impact of using an upper bound as first incumbent. It can also help to get insight whether the hard part of the CRP is to find an optimal solution or to prove optimality of a given optimal solution. We can see that for instances with $T' \leqslant 4$, a majority of instances are solved optimally by the upper bound and the average gap is always less than 1. Table 3.2 shows CRP-I solves all these instances optimally within a few seconds. This suggests that providing the upper bound as an incumbent is beneficial for the integer program and the proof of optimality of this incumbent seems to be relatively fast. For larger classes, there are many fewer instances for which the upper bound is optimal, hence the solution time for these instances increases dramatically. This suggests that the hard part for larger problems is to be find an optimal solution. Once found, CRP-I seems to be quite efficient in proving the optimality of the solution.

Comparison of CRP-I with solutions from [20, 18, 42, 66]

We present Table 3.5 as in [20, 18] to ease the comparison between our solution and theirs. First, the optimal values are all identical, confirming the accuracy of our model. Secondly, CRP-I clearly outperforms the corrected integer programs BRP-II* and BRP2ci in terms of computation time. In Table 3.6, we compare BRP2ci and CRP-I on three other problem classes provided in [18].

Most importantly, we compare CRP-I with the branch-and-bound approach in [20]. For all instances in classes other than $(4, 6)$, even though the branch-and-bound approach is faster, CRP-I still solves instances in less than a second, hence we claim that the difference between these two solution times is minor. However for the hardest problem class in this table, i.e., $(4, 6)$, CRP-I solves all instances in less than 10 seconds, while branch-and-bound requires more than 17 seconds to solve one instance. Expósito-Izquierdo et al. [20] mention that the exact branch-and-bound approach

(T', S)	Instance	Optimal	time (sec.)				
			CRP-I	BRP-II*	B&B	BRP2ci	CC_{15}&PDB_0
$(3, 3)$	1	6	$*$	1.18	0.007	0.11	
	2	5	$*$	1.39	0.007	0.11	
	3	2	$*$	1.00	0.006	0.08	
	4	4	$*$	1.09	0.007	0.09	
	5	1	$*$	1.06	0.007	0.06	
	Avg	3.6	$*$	3.6	0.007	0.09	
$(3, 4)$	1	5	0.014	4.76	0.008	0.28	
	2	3	$*$	18.39	0.007	0.22	
	3	7	$*$	11.71	0.006	0.34	
	4	5	$*$	16.06	0.007	0.26	
	5	6	$*$	18.04	0.007	0.27	
	Avg	5.2	0.014	13.79	0.008	0.274	
$(3, 5)$	1	6	$*$	83.09	0.010	0.72	
	2	7	$*$	75.95	0.007	1.2	
	3	8	$*$	100.71	0.013	0.91	
	4	6	$*$	95.31	0.008	0.8	
	5	9	0.097	65.32	0.026	1.59	
	Avg	7.2	0.097	84.11	0.013	1.044	
$(3, 6)$	1	11	0.397	124.11	0.086	5.35	5.26
	2	7	$*$	113.29	0.008	1.59	0
	3	11	$*$	89.06	0.019	3.09	2.01
	4	7	$*$	93.12	0.016	1.78	0.03
	5	4	$*$	96.50	0.007	1.23	0
	Avg	8.0	0.397	103.22	0.027	2.608	1.46
$(3, 7)$	1	7	$*$	182.14	0.009	2.89	0.08
	2	10	$*$	284.29	0.011	3.29	0.6
	3	9	$*$	119.20	0.011	3.34	1.98
	4	8	$*$	472.32	0.010	3.35	0.3
	5	12	$*$	277.97	0.038	7.97	191.6
	Avg	9.2	$*$	267.26	0.016	4.168	38.91
$(3, 8)$	1	8	0.508	84.66	0.021	7.25	0.75
	2	10	$*$	14403.33	0.055	9.31	8.65
	3	9	$*$	8298.85	0.012	6.57	0.66
	4	10	$*$	250.33	0.013	11.03	7.24
	5	13	$*$	6384.65	0.022	11.06	13.53
	Avg	10.0	0.508	5884.36	0.025	9.044	6.166
$(4, 4)$	1	10	$*$	71.96	0.017	1.25	0.08
	2	10	0.036	228.91	0.025	0.95	0
	3	10	0.077	71.99	0.009	1.56	0.03
	4	7	$*$	65.25	0.008	0.78	0
	5	9	0.081	89.36	0.013	1.15	0.04
	Avg	9.2	0.065	105.49	0.014	1.138	0.03
$(4, 5)$	1	16	0.543	3544.86	0.540	61.62	21.08
	2	10	0.229	326.54	0.043	5.27	0.15
	3	13	0.261	1023.98	0.018	8.28	2.6
	4	8	$*$	119.86	0.014	2.96	0.02
	5	16	0.763	2656.67	0.579	81.26	93.4
	Avg	12.6	0.449	1534.38	0.239	31.878	23.45
$(4, 6)$	1	17	8.679	4077.08	17.476	n.a.	64.73
	2	8	0.541	18985.03	0.030	5.23	0.02
	3	13	$*$	1706.75	0.069	12.86	31.9
	4	14	0.828	2376.86	0.315	23.4	59.53
	5	15	0.641	8564.20	0.076	45.22	107.15
	Avg	13.4	2.672	7141.98	3.593	n.a.	31.24

Table 3.5: Comparison between our formulation CRP-I, BRP-II* and branch-and-bound (B&B) from [20], BRP2ci from [18] and CC_{15}&PDB_0 from [42] on a subset of instances from [9]. Time limits (BRP-II*: 1 day; BRP2ci: 900 seconds) and instances not solved optimally are noted by n.a.. Times are given in seconds. $*$ indicates that the instance is trivial.

(T', S)	Instance	Optimal	time (sec.)	
			CRP-I	BRP2ci
$(4, 7)$	1	17	2.995	44.1
	2	18	1.245	20.47
	3	13	$*$	18.95
	4	16	2.31	49.78
	5	16	2.16	85.75
	Avg	16	2.178	43.81
$(5, 4)$	1	15	0.295	118.2
	2	19	1.06	185.36
	3	15	0.32	5.9
	4	12	0.17	5.01
	5	17	0.652	118.16
	Avg	15.6	0.499	86.526
$(5, 5)$	1	22	6.079	n.a.
	2	16	0.422	22.06
	3	22	25.542	n.a.
	4	21	5.538	n.a.
	5	16	0.837	46.3
	Avg	19.4	7.684	n.a.

Table 3.6: Further comparison between our formulation CRP-I and BRP2ci from [18] on an extended subset of instances from [9]. Time limit (BRP2ci: 900 seconds) and instances not solved optimally are noted by n.a. $*$ indicates that the instance is trivial.

could start to be intractable for this problem size and propose faster heuristic solutions by varying a parameter in their approach.

We also tested the B&B solution from [66] on instances of Tables 3.5 and 3.6, using the code in C provided on the author's website. The bounds used in [66] seem to be tighter, allowing their algorithm to solve all instances in less than 0.1 seconds. A good future direction for our work could be the incorporation of these bounds into our approach. While the B&B approach in [66] is faster, our approach is sufficiently fast for all cases tested in Tables 3.5 and 3.6 (required less than .5 minutes) to allow for real-time decision making. Moreover, one advantage of integer programs (including our method) is that its efficiency relies significantly on the performance of integer programming solvers. Thus, in light of their recent dramatic improvement, if solvers continue to rapidly improve in the coming years, our method will also improve its computational tractability with no change required.

3.4 An Average-Case Asymptotic Analysis of the CRP

In this section, for the sake of the analysis, we consider that we are given a configuration with S stacks, T tiers; Initially C containers are stored in the configuration with exactly h containers in each stack, where $h \leqslant T-1$, so $C = h \times S$. We denote such a configuration $B_{h,S}$. For configuration $B_{h,S}$, we denote the minimum number of relocations by $z_{opt}(B_{h,S})$. We focus on an average case analysis when the number of stacks grows asymptotically. In our model, since $C = h \times S$, when S grows to infinity, C also grows to infinity.

Before stating the main result in Section 3.4.2, we first provide four main ingredients in the next section: the notion of an uniformly random configuration, the simple lower bound $b(.)$ on the minimum number of relocations introduced in [38], a heuristic developped in [10] that performs well on average in large configurations, and the notion of "special" stacks.

3.4.1 Background

Uniformly random configuration

We view a configuration as an array of $T \times S$ slots (see Figure 3-1). The slots are numbered from bottom to top, and left to right from 1 to $T \times S$. The goal is to generate a configuration $B_{h,S}$ with uniform probability, meaning each container is equally likely to be anywhere in the configuration, with the restriction that there are h containers per stack. We first generate a uniformly random permutation of $\{1, \ldots, h \times S\}$ called π. Then we assign a slot for each container with the following relation: $B_{h,S}(i,j) = \pi(h \times (j-1) + i)$ for $i \leqslant h$ and $B_{h,S}(i,j) = 0$ for $i \geqslant h+1$. One can see that each configuration is generated with probability $1/C!$. There is a one to one mapping between configurations with S stacks and permutations of $\{1, \ldots, h \times S\}$, denoted by $\mathcal{S}_{h \times S}$. Finally, we denote the expectation of random variable X over this uniform distribution by $\mathbb{E}_{h,S}[X]$.

The counting lower bound b(.)

This bound was introduced in [38] (also introduced in Section 3.3.1) and it is based on the following simple observation. In the initial configuration, if a container is blocking, then it must be relocated at least once. Thus we count the number of blocking containers in $B_{h,S}$, we denote it as $b(B_{h,S})$, and we have $z_{opt}(B_{h,S}) \geqslant b(B_{h,S})$. Note that if a container blocks more than one container, it is counted only once. In Lemma 2, we give an explicit formula for the expectation of $b(.)$ under the uniform distribution.

Lemma 2. *Let $S, h \in \mathbb{N}$ and $b(.)$ be the counting lower bound defined above, we have*

$$\mathbb{E}_{h,S}\left[b(B_{h,S})\right] = \alpha_h \times S \tag{3.16}$$

where $\alpha_h = h - \sum_{i=1}^{h} 1/i$ is the expected number of blocking containers in one stack.

Observation 1. *Note that α_h only depends on h.*

Proof. Let $b^i(B_{h,S})$ be the number of blocking containers in stack i. By the linearity of expectation, we have $\mathbb{E}_{h,S}\left[b(B_{h,S})\right] = \mathbb{E}_{h,S}\left[\sum_{i=1}^{S} b^i(B_{h,S})\right] = \sum_{i=1}^{S} \mathbb{E}_{h,S}\left[b^i(B_{h,S})\right] = \alpha_h \times S$, where $\alpha_h = \mathbb{E}_{h,S}\left[b^1(B_{h,S})\right] = \mathbb{E}_{h,1}\left[b(B_{h,1})\right]$. This relies on the fact that each stack is identically distributed.

Now let us compute α_h. It is clear that $\alpha_1 = 0$. For $h \geqslant 2$, by conditioning on the event that the topmost container is the smallest number in the stack or not, we obtain the recursive equation $\alpha_h = \alpha_{h-1} + (h-1)/h$. Finally by induction we have $\alpha_h = h - \sum_{i=1}^{h} 1/i$ which completes the proof. □

The heuristic MinMax [10]

Suppose n is the target container located in stack s, and r is the topmost blocking container in s. For convenience, we denote by $\min(s_i)$ the minimum of stack s_i (note that $\min(s_i) = C + 1$ if s_i is empty). MinMax uses the following rule to determine $s^* \neq s$, the stack where r should be relocated to. If there is a stack s_i with $|s_i| < T$, where $\min(s_i)$ is greater than r, then MinMax chooses such a stack where $\min(s_i)$

is minimized, since stacks with larger minimums can be useful for larger blocking containers (as r will never be relocated again, we say this relocation of r is a "good" move). If there is no stack satisfying $\min(s_i) > r$ (any relocation of r can only result in a "bad" move), then MinMax chooses the stack where $\min(s_i)$ is maximized to delay the next unavoidable relocation of r as much as possible. We will refer to this heuristic as heuristic MinMax and denote its number of relocations by $z_M(B_{h,S})$. Notice that $z_{opt}(B_{h,S}) \leqslant z_M(B_{h,S})$. Finally we state the following simple fact, which does not require a formal proof:

Observation 2. *For any configuration B with S stacks and at most S containers, we have*

$$b(B) = z_{opt}(B) = z_M(B). \tag{3.17}$$

Definition of "special" stacks.

For $h, S \in \mathbb{N}$, a stack in $B_{h,S}$ is called "special" if all of its containers belong to the S highest. Given this definition, a stack in $B_{h,S+1}$ is "special" if all containers belong to the $S+1$ highest or equivalently, if each of its containers has index at least $\omega_{h,S} = (h-1)(S+1)+1$. We will also consider the following event:

$$\Omega_{h,S} = \{B_{h,S+1} \textit{ has at least 1 "special" stack}\}. \tag{3.18}$$

Lemma 3 states that the event $\Omega_{h,S}$ has a probability that increases exponentially fast to 1 as a function of S. The proof of Lemma 3 can be found in Appendix A.2.

Lemma 3. *Let $h, S \in \mathbb{N}$ such that $S \geqslant h+1$ and $\Omega_{h,S}$ be the event defined by equation (3.18), then we have*

$$\mathbb{P}(\overline{\Omega_{h,S}}) \leqslant e^{-\theta_h(S+1)}, \tag{3.19}$$

where

$$\theta_h = \frac{1}{8h}\left(\frac{2}{h(h+1)}\right)^{2h} > 0. \tag{3.20}$$

3.4.2 An Average-Case Asymptotic Analysis of CRP

The major result of this section states that when the number of stacks increases to infinity, the expected optimal number of relocations is asymptotically proportional to the expected number of blocking containers.

Theorem 1. *Let $b(.)$ be the counting lower bound and $z_{opt}(.)$ be the optimal number of relocations defined in section 3.1. Then for $h, S \in \mathbb{N}$ such that $S \geqslant h+1$, we have*

$$1 \leqslant \frac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} \leqslant f_h(S) \tag{3.21}$$

where

$$f_h(S) = 1 + \frac{K_h}{S} \underset{C \to \infty}{\to} 1 \tag{3.22}$$

where K_h is a constant defined by equation (3.28).

Proof. The basic intuition is that, as the number of stacks grows, for any blocking container, we can find a "good" stack with high probability. This implies that with high probability, each container is only relocated once. More formally, as S grows, with high probability $\mathbb{E}_{h,S+1}[z_{opt}(B_{h,S+1})] - \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]$ is exactly α_h. Therefore, for large enough S, $\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]$ essentially behaves like $\alpha_h \times S$, which is equal to $\mathbb{E}_{h,S}[b(B_{h,S})]$ (according to Lemma 2).

Since for all configurations $B_{h,S}$, $z_{opt}(B_{h,S}) \geqslant b(B_{h,S})$ then $\dfrac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} \geqslant 1$.

Moreover, we have:

$$\begin{aligned}
\frac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} &= 1 + \frac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] - \mathbb{E}_{h,S}\left[b(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} \\
&= 1 + \frac{1}{\alpha_h S}\left(\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] - \alpha_h S\right) \\
&= 1 + \frac{g_h(S)}{\alpha_h S},
\end{aligned} \tag{3.23}$$

where

$$g_h(S) = \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] - \alpha_h S. \tag{3.24}$$

Now we claim that there exists a constant $\theta_h > 0$ (defined in equation (3.20) such

that:

$$\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1})\right] \leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + \alpha_h + h(T-1)(S+1)e^{-\theta_h(S+1)},\ \forall S \geqslant h+1 \quad (3.25)$$

Equation (3.25) studies how $\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]$ evolves and states that it increases almost linearly in α_h which shows that the function $g_h(.)$ is essentially bounded. Before proving equation (3.25), we conclude the proof of Theorem 1. Using equation (3.25), we have for all $S \geqslant h+1$:

$$\begin{aligned}
&\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1})\right] \leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + \alpha_h + h(T-1)(S+1)e^{-\theta_h(S+1)} \\
\implies &\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1})\right] - \alpha_h(S+1) \leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] - \alpha_h S + h(T-1)(S+1)e^{-\theta_h(S+1)} \\
\implies &g_h(S+1) \leqslant g_h(S) + h(T-1)(S+1)e^{-\theta_h(S+1)} \\
\implies &g_h(S) \leqslant g_h(h+1) + h(T-1)\sum_{i=h+2}^{S}\left(ie^{-\theta_h i}\right) \\
\implies &g_h(S) \leqslant g_h(h+1) + h(T-1)\sum_{i=1}^{\infty}\left(ie^{-\theta_h i}\right) \\
\implies &g_h(S) \leqslant g_h(h+1) + \frac{e^{\theta_h}h(T-1)}{\left(e^{\theta_h}-1\right)^2} = K'_h. \qquad (3.26)
\end{aligned}$$

Therefore using equations (3.23) and (3.26), we have

$$\frac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} \leqslant 1 + \frac{K_h}{S} = f_h(S), \qquad (3.27)$$

where

$$K_h = \frac{K'_h}{\alpha_h} = \frac{g_h(h+1) + \dfrac{e^{\theta_h}h(T-1)}{\left(e^{\theta_h}-1\right)^2}}{\alpha_h}. \qquad (3.28)$$

Now let us prove equation (3.25). Recall that a stack is defined to be "special" if none of its containers are smaller than $\omega_{h,S} = (h-1)(S+1)+1$ and that $\Omega_{h,S} = \{B_{h,S+1}$ *has at least one "special" stack*$\}$.

The intuition is the following: the probability of having a "special" stack grows quickly to 1 as a function of S, implying that the event $\Omega_{h,S}$ happens with high

probability. Now, conditioned on $\Omega_{h,S}$, we more easily express the difference between configurations of size $S+1$ and S in the following way. We claim that

$$\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1})|\Omega_{h,S}\right] \leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + \alpha_h. \tag{3.29}$$

Let $B_{h,S+1}$ be a given configuration with $S+1$ stacks that verifies $\Omega_{h,S}$. Since stacks in configurations can be interchanged, we suppose that a "special" stack is the first (leftmost) stack of the configuration. We also denote $n_1, n_2, \ldots, n_h$ the containers of the first stack. We know that $n_1, n_2, \ldots, n_h \geqslant \omega_{h,S}$ and $n_1 \neq n_2 \neq \ldots \neq n_h$. Finally let $\hat{B}_{h,S}$ be the configuration $B_{h,S+1}$ without its first stack (see Figure 3-3).

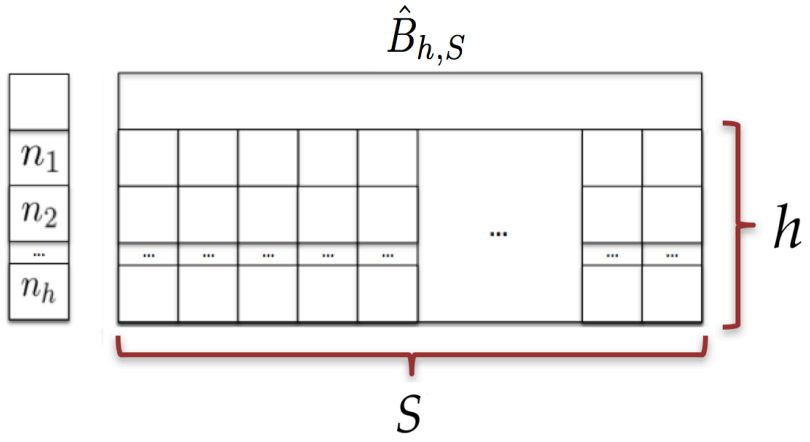


Figure 3-3: Decomposition of the configuration $B_{h,S+1}$ (The right part has S stacks).

First we prove that

$$z_{opt}(B_{h,S+1}) \leqslant z_{opt}(\hat{B}_{h,S}) + b\left(\begin{bmatrix} n_1 \\ \ldots \\ n_h \end{bmatrix}\right). \tag{3.30}$$

To prove equation (3.30), we construct a feasible sequence σ for the configuration of size $S+1$ for which the number of relocations is equal to the right side of equation (3.30). Let $\sigma_{opt}(\hat{B}_{h,S})$ the optimal sequence for $\hat{B}_{h,S}$, $t' = \min\{n_1, \ldots, n_h\}$ be the first time step when the target container in $\sigma_{opt}(\hat{B}_{h,S})$ is larger than $\min\{n_1, n_2, \ldots, n_h\}$ and $B'_{h,S}$ be the configuration obtained at t' using $\sigma_{opt}(\hat{B}_{h,S})$. Let the first $t'-1$ moves

of σ be the first $t'-1$ moves of $\sigma_{opt}(\hat{B}_{h,S})$. Note that $B'_{h,S}$ has at most $S+1-h$ (which is at most S) containers due to the choice of $\omega_{h,S}$. By Fact 2, the number of relocations performed by $\sigma_{opt}(\hat{B}_{h,S})$ from t' until the end is $b(B'_{h,S})$. Therefore

$$z_{opt}(\hat{B}_{h,S}) = \left\{ \begin{array}{c} \#\ relocations\ up\ to\ t' \\ done\ by\ \sigma_{opt}(\hat{B}_{h,S}) \end{array} \right\} + b(B'_{h,S}) \tag{3.31}$$

After t', we run heuristic MinMax on

$$B'_{h,S+1} = \left(\left[\begin{array}{ccc} n_1 & \dots & n_h \end{array} \right]^T \cup\ B'_{h,S} \right).$$

We claim that z_σ (number of relocations performed by the feasible sequence σ constructed above) is exactly the right side of equation (3.30). There are at most $S+1$ containers in $B'_{h,S+1}$, therefore using Fact 2, we know that if we apply the heuristic MinMax to this configuration, then the number of relocations done by MinMax is $b(B'_{h,S+1}) = b(B'_{h,S}) + b\left(\left[\begin{array}{ccc} n_1 & \dots & n_h \end{array} \right]^T \right)$. Therefore $z_{opt}(B_{h,S+1}) \leqslant z_\sigma(B_{h,S+1})$ and

$$z_\sigma(B_{h,S+1}) = \left\{ \begin{array}{c} \#\ relocations\ up\ to\ t' \\ done\ by\ \sigma_{opt}(\hat{B}_{h,S}) \end{array} \right\} + b(B'_{h,S}) + b\left(\left[\begin{array}{c} n_1 \\ \dots \\ n_h \end{array} \right] \right) \tag{3.32}$$

which gives us

$$z_{opt}(B_{h,S+1}) \leqslant z_{opt}(\hat{B}_{h,S}) + b\left(\left[\begin{array}{c} n_1 \\ \dots \\ n_h \end{array} \right] \right),$$

and proves equation (3.30).

Now we can take the expectation from both sides of equation (3.30) over a uniform distribution of the rest of the $h \times S$ containers that are not in the first stack. We claim that the first term on the right hand-side of equation (3.30) is exactly $\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]$. For any configuration that appears in $\hat{B}_{h,S}$ we can map it to a unique configuration

$B_{h,S}$ where all containers are between 1 and hS, and vice versa. Thus,

$$\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S+1}) \;\middle|\; \begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \right] \leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + b\left(\begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \right). \tag{3.33}$$

Next, we take the expectation of both sides of equation (3.33) over possible first stacks, which is a "special" stack. Now notice that if $B_{h,S+1}$ is generated uniformly in the sets of configurations of size $S+1$, then conditioned on $\Omega_{h,S}$, the probability of having a certain stack $[n_1, \dots, n_h]^T$ is identical for any $n_1 \neq \dots \neq n_h \geqslant \omega_{h,S}$ and it is given by

$$\mathbb{P}\left(\begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \;\middle|\; \Omega_{h,S} \right) = \frac{(S+1-h)!}{(S+1)!} = \frac{1}{\binom{S+1}{h} h!}.$$

Therefore we can write:

$$\begin{aligned}
&\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \Omega_{h,S}\right] \\
&= \sum_{\substack{(n_1,\dots,n_h) \\ n_i \neq n_j \\ n_i \geqslant \omega_{h,S}}} \left(\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S+1}) \;\middle|\; \begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix}, \Omega_{h,S} \right] \times \mathbb{P}\left(\begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \;\middle|\; \Omega_{h,S} \right) \right) \qquad (3.34) \\
&= \sum_{\substack{(n_1,\dots,n_h) \\ n_i \neq n_j \\ n_i \geqslant \omega_{h,S}}} \left(\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S+1}) \;\middle|\; \begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \right] \times \mathbb{P}\left(\begin{bmatrix} n_1 \\ \dots \\ n_h \end{bmatrix} \;\middle|\; \Omega_{h,S} \right) \right) \qquad (3.35)
\end{aligned}$$

$$\leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] \sum_{\substack{(n_1,\ldots,n_h)\\ n_i \neq n_j\\ n_i \geqslant \omega_{h,S}}} \mathbb{P}\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right] \,\middle|\, \Omega_{h,S}\right)$$

$$+ \sum_{\substack{(n_1,\ldots,n_h)\\ n_i \neq n_j\\ n_i \geqslant \omega_{h,S}}} b\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right]\right) \times \mathbb{P}\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right] \,\middle|\, \Omega_{h,S}\right)$$

$$\leqslant \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + \sum_{\substack{(n_1,\ldots,n_h)\\ n_i \neq n_j\\ n_i \geqslant \omega_{h,S}}} b\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right]\right) \times \mathbb{P}\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right] \,\middle|\, \Omega_{h,S}\right) \tag{3.36}$$

The equality between (3.34) and (3.35) comes from the fact that if we know that $B_{h,S+1}$ has a "special" stack, then we do not need to condition on $\Omega_{h,S}$. Equation (3.36) uses the fact that $\sum_{\substack{n_1 \neq \ldots \neq n_h\\ n_i \geqslant \omega_{h,S}}} \mathbb{P}\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right] \,\middle|\, \Omega_{h,S}\right) = 1$. Note that, given any $(n_1, \ldots, n_h)$ such that $n_i \neq n_j$, we have

$$\mathbb{E}_{h,1}\left[b\left(\left[\begin{array}{c} n_1 \\ \ldots \\ n_h \end{array}\right]\right)\right] = \alpha_h,$$

where the expectation is over a random order of $(n_1, \ldots, n_h)$. This is true regardless of the set $(n_1, \ldots, n_h)$ that is drawn from (See Fact 1). This implies that the second term in the right hand side of equation (3.36) is equal to α_h; Therefore, we get equation (3.29). Now we want to focus on the event $\overline{\Omega_{h,S}}$. We give an upper bound on $\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \overline{\Omega_{h,S}}\right]$. For any configuration, to retrieve one container, we need at most $T-1$ relocations (since at most $T-1$ containers are blocking it), thus for any configuration, the optimal number of relocations is at most $T-1$ times the number of containers ($h(S+1)$) which gives us $h(T-1)(S+1)$ as an upper bound

on the optimal number of relocations. We use this universal bound to get

$$\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \overline{\Omega_{h,S}}\right] \leqslant h(T-1)(S+1). \tag{3.37}$$

Finally using Lemma 3, we have

$$\begin{aligned}
\mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1})\right] &= \mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \Omega_{h,S}\right]\mathbb{P}(\Omega_{h,S}) + \mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \overline{\Omega_{h,S}}\right]\mathbb{P}(\overline{\Omega_{h,S}}) \\
&\leqslant \mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \Omega_{h,S}\right] + \mathbb{E}_{h,S+1}\left[z_{opt}(B_{h,S+1}) \mid \overline{\Omega_{h,S}}\right]\ e^{-\theta_h(S+1)} \\
&\leqslant\ \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right] + \alpha_h + h(T-1)(S+1)e^{-\theta_h(S+1)}
\end{aligned}$$

which proves equation (3.25), hence completes the proof of Theorem 1. □

In the next corollary, we show that the optimal solution of the unrestricted CRP has a similar asymptotic behavior. We remind that the unrestricted CRP refers to the problem where we can also relocate non-blocking containers. The proof is trivial since by definition $b(B_{h,S}) \leqslant z_{unr}(B_{h,S}) \leqslant z_{opt}(B_{h,S})$.

Corollary 1. *Let $z_{unr}(B_{h,S})$ be the optimal number of relocations for the unrestricted CRP. For $S \geqslant h+1$, we have*

$$1 \leqslant \frac{\mathbb{E}_{h,S}\left[z_{unr}(B_{h,S})\right]}{\alpha_h S} \leqslant f_h(S),$$

where f_h is the function defined in Theorem 1.

Experimental results on the efficiency of heuristic H

Theorem 1 gives insights on how the expected optimal solution of the CRP behaves asymptotically on random configurations. To give more insights on CRP, we show experimentally that the same result holds for heuristic MinMax, i.e., the ratio of $\mathbb{E}_{h,S}\left[z_M(B_{h,S})\right]$ and $\mathbb{E}_{h,S}\left[b(B_{h,S})\right]$ converges to 1 as S goes to infinity. We take $h = T-1 = 4$ and for each size S, we compute both expectations over a million instances generated uniformly, take their ratio and plot the result in Figure 3-4. Notice that

we have

$$1 \leqslant \frac{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]} \leqslant \frac{\mathbb{E}_{h,S}\left[z_M(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]},$$

so Figure 3-4 also shows experimentally that Theorem 1 holds.

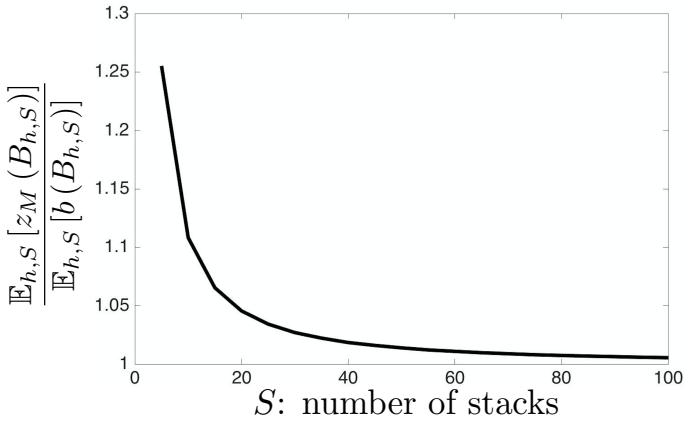


Figure 3-4: Simulation of the convergence of the ratio.

First, note that Figure 3-4 implies that the relative gap between heuristic MinMax and b shrinks to 0 as S increases. Moreover we have

$$\frac{\mathbb{E}_{h,S}\left[z_M(B_{h,S})\right] - \mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[z_{opt}(B_{h,S})\right]} \leqslant \frac{\mathbb{E}_{h,S}\left[z_M(B_{h,S})\right] - \mathbb{E}_{h,S}\left[b(B_{h,S})\right]}{\mathbb{E}_{h,S}\left[b(B_{h,S})\right]},$$

and thus the relative gap of MinMax with optimality also converges to 0 as S grows to infinity.

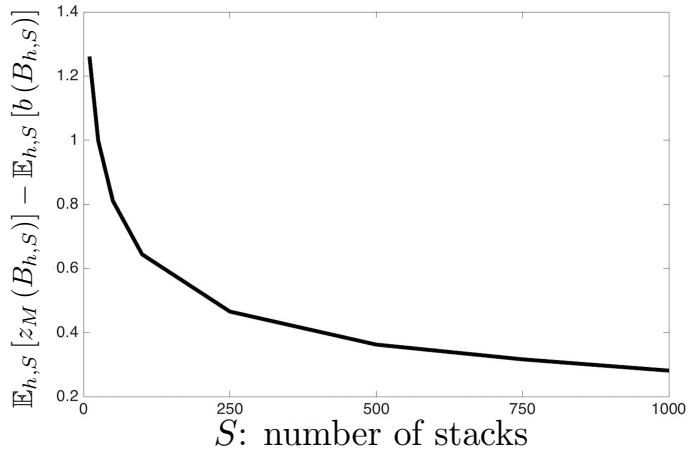


Figure 3-5: Simulation of the convergence of the difference.

In the proof of Theorem 1, we also study the function $g_h(S) = \mathbb{E}_{h,S}[z_{opt}(B_{h,S})] - \mathbb{E}_{h,S}[b(B_{h,S})]$. Note that $g_h(S) \leqslant \mathbb{E}_{h,S}[z_M(B_{h,S})] - \mathbb{E}_{h,S}[b(B_{h,S})]$ where the right-hand side of the inequality is the function plotted in Figure 3-5. The plot shows that $g_h(S) \leqslant 1.25$ for all S, meaning that $g_h(S)$ is bounded as we proved in Theorem 1. Moreover, the plot implies that heuristic MinMax is on average at most 1.25 away from the optimal solution, so heuristic MinMax is relatively more efficient in the case of large configurations. Intuitively, the probability of having a good stack converges to 1, as we increase the number of stacks; hence the problem tends to become easier as S grows.

Finally, in the proof, we note that the rate of convergence of the minimum z_{opt} to b is at least $1/S$. Interestingly, we can infer from Figure 3-4. that the rate of convergence of the ratio for heuristic MinMax is also proportional to $1/S$.

Chapter 4

The Stochastic Container Relocation Problem

4.1 Contributions

The contributions of this chapter are (each contribution corresponds to one section of this chapter):

1. A new stochastic model, referred to as the *batch model*. This new model uses the same probability distribution as the online model from [86]. However, the two models differ in the way each reveals new information on the retrieval order. The batch model is motivated, described, and compared with the online model.

2. Lower and upper bounds for the SCRP. We derive a **new family of lower bounds** for which we show theoretical properties. Furthermore, we develop **two new fast and efficient heuristics**.

3. **A novel optimal algorithm scheme based on decision trees and pruning strategies referred to as *Pruning-Best-First-Search* (*PBFS*)**, which takes advantage of the properties of the aforementioned lower bounds. The algorithm is explained with pseudocode in Algorithm 2.

4. **A second novel algorithm tuned for the case of larger batches referred**

to as *PBFS-Approximate* (*PBFSA*). We build upon *PBFS* and derive a sampling strategy resulting in an approximate algorithm with an expected error that we bound theoretically. The pseudocode of the second algorithm is presented in Algorithm 3.

5. **Extensive computational experiments using an existing set of instances**. The first experiment exhibits the efficiency of *PBFS*, our lower bounds and two new heuristics for the batch model based on existing instances, presented in [43], where batches of containers are small (2 containers per batch on average). The second experiment illustrates the advantage of using *PBFSA* when batches of containers are larger, based on instances obtained by modifying the existing set. In addition, most of our techniques including lower bounds, heuristics, and the *PBFS* algorithm also apply to the online model. The third experiment shows that, in this model, *PBFS* outperforms the algorithm introduced in [43] in the sense that it is faster for instances that Ku and Arthanari [43] could solve, and it can solve problems of larger size. Furthermore, our two new heuristics also outperform the best existing heuristic (ERI) for the online model. Finally, the last experiment is used to test the conjecture about the optimality of the leveling heuristic in the special case of the online model with a unique batch of containers.

4.2 Problem Description

4.2.1 Motivation

Before stating the general assumptions of the batch model, let us motivate our problem using a typical example. We consider a port with a truck appointment system (TAS) offering 30-minute time windows during which truck drivers who want to retrieve a container can register to arrive at the port. For the sake of the example, we consider the time window between 9:00 a.m. and 9:30 a.m. Multiple trucks can be registered in this time window: in this example, presented in Figure 4-1, 3 trucks

(designated i_1, i_4 and i_6) are registered for this time window. We assume that **all 3 trucks arrive on time** (between 9:00 a.m. and 9:30 a.m.) and that their containers (similarly designated i_1, i_4 and i_6) form a batch to be retrieved. We display the configuration of interest in Figure 4-2 (3 tiers, 3 stacks and 6 containers).

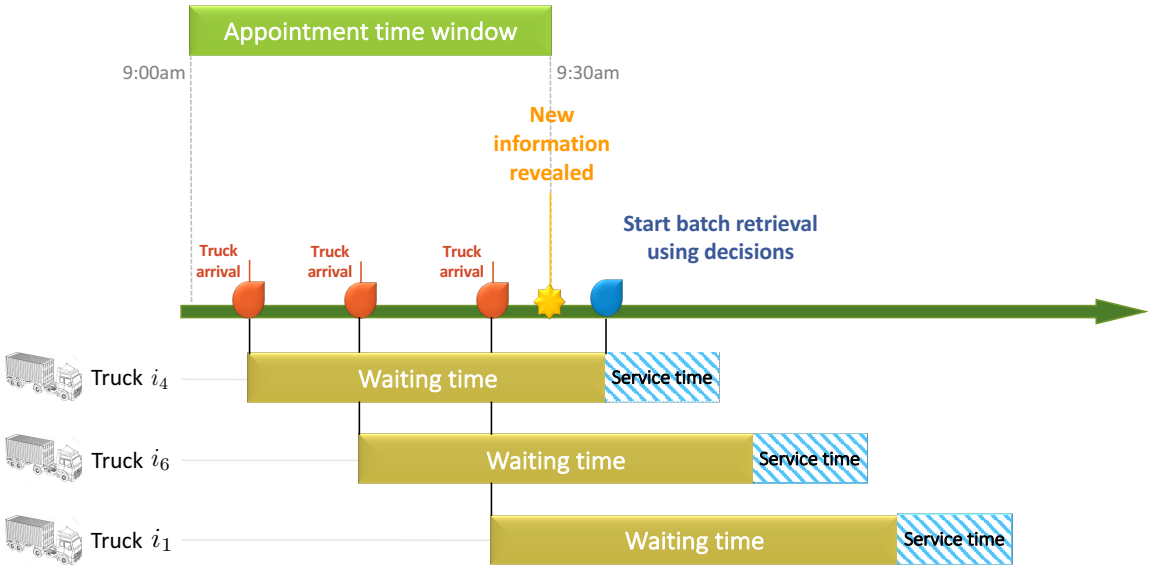


Figure 4-1: Timeline of events for the batch model with three trucks.

We assume that trucks arrive randomly within the time window, so **each truck arrival order is equally likely to happen**. In this example, there are 6 potential arrival orders, each with 1/6 chance of occurring.

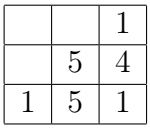
At 9:00 a.m., none of the 3 trucks has arrived and **their relative retrieval order is unknown**. Consequently, these 3 containers are all labeled 1 in Figure 4-2a. In Figure 4-2b, the IDs of all containers and their locations are depicted.

Between 9:00 a.m. and 9:30 a.m., trucks arrive in a particular order (e.g., Truck i_4 first, then i_6, and i_1 last). In busy terminals, trucks typically queue up as they wait to be served. Their place in line is based on their arrival order, so the port operator generally retrieves containers based on the arrival order. Processing in this way, **on a first-come first-served basis**, avoids issues with truck unions and maintains fairness among drivers. Consequently, we take the retrieval order to be exogenously determined and we do not consider it a potential decision for port operators.

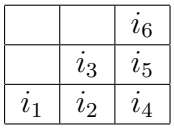
To provide a specified level of service to the truck drivers, the terminal operator

often sets a target average waiting time. If the appointment time window is about the same as or shorter than the target average waiting time, the operator has information about the retrieval order of containers in the batch before the retrieval of those containers must begin to meet the target waiting time. Given these conditions, we make the simplifying assumption in this work that the retrieval of a batch begins at the end of the appointment time window associated with the batch, and **the retrieval order of all containers in the batch is known at the moment the retrieval of the batch commences**. In our example, the target average waiting time is 30 minutes. At 9:30 a.m., the retrieval order of the batch (i_4, i_6, i_1) is known and the retrieval of the batch commences soon after. The updated information is depicted in the configuration of Figure 4-2c.

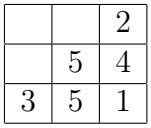
The assumption that containers to be retrieved are revealed on a batch basis models the reality that port operators typically know information about all the containers in the same batch before starting to retrieve them. This is especially true for busy ports that have a TAS. Moreover, we assume that no information about future batches is available when making decisions for the current batch. Similar modeling assumptions have been made in previous work (see [86, 43]).



4-2a Before any truck has arrived (9:00 a.m.).



4-2b IDs to match containers with trucks.



4-2c Before the first container gets retrieved (9:30 a.m.).

Figure 4-2: SCRP example. The left configuration is the input to our problem. The configuration in the middle denotes each container with an ID i_l where $l = 1, \ldots, 6$. The configuration on the right denotes the order of the first batch after it is revealed.

The general assumptions we apply to our model are formally stated in Section 4.2.2 (A_5^* and A_6^*) and result in the *batch model*, *the main focus of this chapter*. The goal of the SCRP is to find a sequence of moves minimizing the expected number of relocations needed to empty the initial configuration.

Labels in Figures 4-2a and 4-2c are defined such that two containers have the

same label only if they are in the same batch and their relative order is yet to be revealed. In our example, since Container i_5 is the only container in the second batch and is retrieved after the first 3 containers, it is necessarily the fourth container to be retrieved (thus labeled 4). Containers i_2 and i_3 are labeled 5 and when their relative order is revealed, one will be labeled 5 and the other 6. We mention that

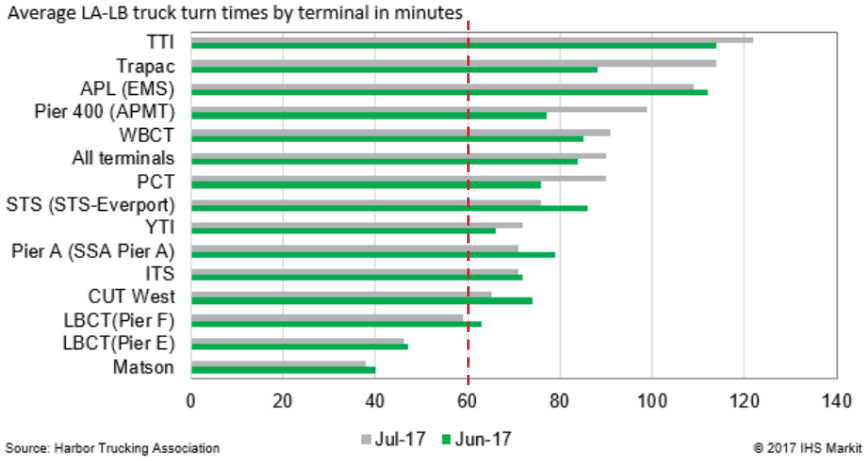


Figure 4-3: Average truck turn times in minutes by terminal at Los Angeles-Long Beach port in June and July 2017 (source: JOC.com). The dashed line shows the length of a time window (60 minutes) in the truck appointment system.

these assumptions apply to an increasing number of port terminals. For example, Figure 4-3 presents the average truck turn time in minutes for all 15 terminals of Los Angeles-Long Beach port (the largest U.S. port for containers) in June and July 2017. In many of these terminals, a truck appointment system with 60-minutes time windows (see [14]) has been implemented, and Figure 4-3 shows that most terminals have an average truck turn time longer than 60 minutes.

The online model. The main distinguishing difference between the batch model and the online model introduced in [86] is the information revelation process. The online model disregards any within-batch information available when it plans the moves to retrieve a container. Hence **new information is revealed one container at a time.** The online model applies specially to less busy ports where the waiting time is significantly shorter than the appointment time windows. In this case, because there is a limited number of trucks waiting, limited information about the future is

known. We show through Lemma 4 that ignoring information (if available) results in a potential loss of operational efficiency as measured by the expected number of relocations. In addition, most of our batch-based approaches also apply to the online model, and we provide numerical results based on it in Section 4.7.

4.2.2 Assumptions, Notations, and Formulation

To define our problem as a multistage stochastic optimization problem (the number of "stages" is the number of batches), we need to define *a probabilistic model of the container retrieval order, a scheme for revealing new information about this order, and an objective function.*

The batch model. Let us state the assumptions and objective of the stochastic CRP under the batch model. Assumptions A_1^*, A_2^*, A_3^*, and A_4^* are respectively identical to Assumptions A_1, A_2, A_3, and A_4 (see Section 3.2).

> $\mathbf{A_5^*}$: (probabilistic model) Given an ordering of batches, the probability distribution of the retrieval order is such that: 1) **all the containers in a given batch are retrieved before any containers in a later batch**, and 2) **within each batch of containers, the order of the containers is drawn from a uniform random permutation**. This chapter focuses mainly on this latter assumption, but our model can be extended to the more general case of any probability distribution on permutations (not necessarily uniform) that respects the order of batches.
>
> $\mathbf{A_6^*}$: (revelation of new information) For each batch w, **the full relative order of containers from the w^{th} batch (i.e., the specific random permutation drawn) is revealed after all containers in batch 1 through $w-1$ have been retrieved**.

Given these assumptions, we want to find the minimum expected number of relocations to retrieve all containers from a given initial configuration. We refer to this problem as the "*stochastic CRP*." Let us introduce some notations:

- The problem size is given by (T, S, C), respectively the number of tiers, stacks, and containers in the initial configuration.

- The number of batches of containers in the initial configuration is denoted by W. We consider that the batches are ordered from 1 to W.

- For each batch $w \in \{1, \ldots, W\}$, let C_w be the number of containers in w. By definition $\sum_{w=1}^{W} C_w = C$.

- Each container has two attributes:

  - The first attribute, denoted by $(c_l)_{l\in\{1,\ldots,C\}}$, is the label and is defined as follows: initially, containers in batch w are labeled by K_w such that $K_w = 1 + \sum_{u=1}^{w-1} C_u$, where the sum is empty for $w = 1$. Then, for $k \in \{1, \ldots, C\}$, if a container is revealed to be the k^{th} container to be retrieved, its label changes to k. Using this labeling, at any point in the retrieval process, two containers have the same label only if they are in the same batch and their relative order is yet to be revealed.

  - The second attribute is a unique ID denoted by $(i_l)_{l\in\{1,\ldots,C\}}$. This ID is only used to identify uniquely containers in the initial configuration (see Figure 4-2b) and for the sake of clarity of the following probabilistic model. Note that for $l \in \{1, \ldots, C\}$, if Container i_l is in batch w, then $c_l = K_w$ (until the actual retrieval order of i_l is revealed).

- For $k \in \{1, \ldots, C\}$, let ζ_k be a random variable taking values in $(i_l)_{l\in\{1,\ldots,C\}}$, such that $\{\zeta_k = i_l\}$ is the event that Container i_l is the k^{th} container to be retrieved. According to Assumption A_5^*, the distribution of $(\zeta_k)_{k\in\{1,\ldots,C\}}$ is given by

$$\mathbb{P}\left[\zeta_k = i_l\right] = \begin{cases} \frac{1}{C_w} \text{ , if } w = \min\{u \in \{1, \ldots, W\} \mid K_u \geqslant k\} \text{ and } c_l = K_w \\ 0 \text{ , otherwise} \end{cases}$$

In this case, there are a total of $\prod_{w=1}^{W} (C_w!)$ orders with equal probabilities. More generally, we consider the case where the probability of each order within

each batch is not necessarily equally likely. However, we still assume that the batches are ordered, thus $\mathbb{P}[\zeta_k = i_l]$ can be positive only if $w = \min\{u | K_u \geqslant k\}$ and $c_l = K_w$. In the practical case where probabilities are not considered to be uniform, a list of potential retrieval orders and their associated probability is given for each batch of containers (based on historical data), thus $\mathbb{P}[\zeta_k = i_l]$ can easily be inferred from these probabilities.

- An action corresponds to a sequence of relocations to retrieve one container. For $k \in \{1, \ldots, C\}$, we denote the action for the k^{th} retrieval by a_k, and the feasible set of actions is defined according to Assumptions A_2^* and A_3^*.

- For a given batch $w \in \{1, \ldots, W\}$,

  1. let y_w denote the configuration before the retrieval order of containers in batch w is revealed, i.e., before $\zeta_{K_w}, \ldots, \zeta_{K_w+C_w-1}$ are revealed. Note that y_1 corresponds to the initial configuration. We denote x_{K_w} the configuration after the retrieval order of containers in batch w is revealed, and before action a_{K_w} is taken. If $\xrightarrow{\zeta_{K_w},\ldots,\zeta_{K_w+C_w-1}}$ represents the revelation of the random variables $\zeta_{K_w}, \ldots, \zeta_{K_w+C_w-1}$, we can write $y_w \xrightarrow{\zeta_{K_w},\ldots,\zeta_{K_w+C_w-1}} x_{K_w}$.

  2. After the retrieval order of batch w has been revealed, actions to retrieve the revealed containers must be made. If $C_w > 1$, then $\{K_w, \ldots, K_w + C_w - 2\} \neq \emptyset$. In this case, for $k \in \{K_w, \ldots, K_w + C_w - 2\}$, let x_{k+1} be the configuration after applying action a_k to state x_k and before action a_{k+1}. Therefore, if $\xrightarrow{a_k}$ represents the application of action a_k, we have $x_k \xrightarrow{a_k} x_{k+1}$.

  3. The last container to be retrieved in batch w is the $(K_w + C_w - 1)^{th}$ container, thus, according to the previous point, $x_{K_w+C_w-1}$ corresponds to the configuration before $a_{K_w+C_w-1}$ is taken. After this retrieval, the order of the next batch (batch $w+1$) has to be revealed, and according to the first point, the configuration is y_{w+1}. The configuration after retrieving batch W will be empty, thus we define y_{W+1} to be the empty configuration. So

if $\xrightarrow{a_{K_w+C_w-1}}$ represents the application of action $a_{K_w+C_w-1}$, then we have $x_{K_w+C_w-1} \xrightarrow{a_{K_w+C_w-1}} y_{w+1}$.

In summary, we have

$$\forall\, w \in \{1,\ldots,W\}\,, \begin{cases} y_w \xrightarrow{\zeta_{K_w},\ldots,\zeta_{K_w+C_w-1}} x_{K_w} \\ x_k \xrightarrow{a_k} x_{k+1},\ \text{if } C_w > 1\ ,\ \forall\ k \in \{K_w,\ldots,K_w+C_w-2\} \\ x_{K_w+C_w-1} \xrightarrow{a_{K_w+C_w-1}} y_{w+1}. \end{cases}$$

- Let the function $r(.)$ be such that $r(x)$ is number of relocations to retrieve the target container in configuration x. It is also equal to the number of containers blocking the target container. This function is only defined for configurations in which the target container is identified. Specifically, it is defined for $(x_k)_{k=1,\ldots,C}$ (but not for $(y_w)_{w=1,\ldots,W}$). For $k \in \{1,\ldots,C\}$, we refer to $r(x_k)$ as the *immediate cost* for the k^{th} retrieval.

- Let the function $f(.)$ be such that $f(x)$ is the minimum expected number of relocations required to retrieve all containers from configuration x. Typically, $f(x)$ is referred to as the *cost-to-go* function of configuration x. Note that it is well defined for both $(x_k)_{k=1,\ldots,C}$ and $(y_w)_{w=1,\ldots,W}$.

By definition we have:

$$\forall\, w \in \{1,\ldots,W\},\ \begin{cases} f(y_w) = \underset{\zeta_k,\ldots,\zeta_{k+C_w-1}}{\mathbb{E}}[f(x_k)] & ,\ \text{where } k = K_w, \\ f(x_k) = r(x_k) + \min\limits_{a_k}\{f(x_{k+1})\} & ,\ \text{if } C_w > 1 \text{ and } \forall\ k \in \{K_w,\ldots,K_w+C_w-2\}, \\ f(x_k) = r(x_k) + \min\limits_{a_k}\{f(y_{w+1})\} & ,\ \text{where } k = K_w+C_w-1, \end{cases}$$

which can be written as

$$\forall\, w \in \{1,\ldots,W\},\ \begin{cases} f(y_w) = \underset{\zeta_{K_w},\ldots,\zeta_{K_w+C_w-1}}{\mathbb{E}}[f(x_{K_w})]\,, \\ f(x_{K_w}) = \min\limits_{a_{K_w},\ldots,a_{K_w+C_w-1}} \left\{ \left(\sum\limits_{k=K_w}^{K_w+C_w-1} r(x_k) \right) + f(y_{w+1}) \right\}, \end{cases} \tag{4.1}$$

and $f(y_{W+1}) = 0$. Therefore, the SCRP is concerned with finding $f(y_1)$, where by induction:

$$f(y_1) = \underset{\zeta_{K_1},\ldots,\zeta_{K_1+C_1-1}}{\mathbb{E}}\left[\min_{a_{K_1},\ldots,a_{K_1+C_1-1}}\left\{\underset{\zeta_{K_2},\ldots,\zeta_{K_2+C_2-1}}{\mathbb{E}}\left[\ldots\underset{\zeta_{K_W},\ldots,\zeta_{K_W+C_W-1}}{\mathbb{E}}\left[\min_{a_{K_W},\ldots,a_{K_W+C_W-1}}\left\{\sum_{k=1}^{C} r(x_k)\right\}\right]\ldots\right]\right\}\right]. \quad (4.2)$$

The online model. Using our notations, we briefly present the SCRP under the online model. Instead of Assumption A_6^*, the online model assumes that for each retrieval $k \in \{1,\ldots,C\}$, only the next target container is revealed (i.e., ζ_k). Therefore, we consider the states $(y_k^o, x_k^o)_{k=1,\ldots,C}$ defined such that $k \in \{1,\ldots,C\}$, $y_k^o \xrightarrow{\zeta_k} x_k^o \xrightarrow{a_k} y_{k+1}^o$, where y_{C+1}^o is the empty configuration. In this case, if f^o denotes the cost-to-go function, then by definition we have $f^o(y_k^o) = \underset{\zeta_k}{\mathbb{E}}\left[f^o(x_k^o)\right]$ (with $f^o\left(y_{C+1}^o\right) = 0$), and $\forall\, k \in \{1,\ldots,C\},\ f^o(x_k^o) = \min_{a_k}\left\{r(x_k^o) + f^o\left(y_{k+1}^o\right)\right\}$. By induction, the SCRP under the online model is hence concerned with finding:

$$f^o(y_1^o) = \underset{\zeta_1}{\mathbb{E}}\left[\min_{a_1}\left\{\underset{\zeta_2}{\mathbb{E}}\left[\ldots\underset{\zeta_C}{\mathbb{E}}\left[\min_{a_C}\left\{\sum_{k=1}^{C} r(x_k^o)\right\}\right]\ldots\right]\right\}\right].$$

The next lemma compares the batch and the online models theoretically (the proof can be found in the Appendix B). It states that it is beneficial in terms of the expected number of relocations to use the batch model compared to the online model, if the first one applies. Practically, this suggests that the operator should always use available information.

Lemma 4. *Let y be a given initial configuration, then we have*

$$f(y) \leqslant f^o(y).$$

4.3 Decision Trees

Multistage stochastic optimization problems can be solved using decision trees in which *chance nodes* and *decision nodes* typically alternate. A *chance node* embodies the stochasticity of the model, whereas a *decision node* models the possible actions

of the algorithm. In a decision tree for the SCRP, a node represents a configuration. The root node (denoted by 0) is the initial configuration, and the leaf nodes are configurations for which we can compute the cost-to-go function.

In our case, we slightly modify the structure of a typical decision tree, in the sense that chance nodes and decision nodes do not necessarily alternate. A chance node is a configuration for which the target container is not known yet, and information needs to be revealed (note that this only occurs at the beginning of each batch). A decision node is a configuration for which the target container is known. Using our notations, chance nodes correspond to $(y_w)_{w=1,\dots,W}$ and decision nodes correspond to $(x_k)_{k=1,\dots,C}$.

Let n be a node corresponding to a configuration in the decision tree. Thus $f(n)$, the cost-to-go function of n, is defined for all nodes n, and $r(n)$, the immediate cost function of n, is well defined when n is a decision node. We denote by λ_n the level of n in the decision tree, and define it as the number of containers remaining in the configuration. We denote the lowest level of the tree by $\lambda^* = \min_{n \in Tree} \{\lambda_n\}$. It corresponds to the level such that if $\lambda_n = \lambda^*$, $f(n)$ can be computed efficiently (the empty configuration being an obvious candidate with a cost-to-go of 0). Moreover,

- If n is a chance node, then there exists a unique $w \in \{1, \dots, W\}$ such that $\lambda_n = C - K_w + 1$. We denote by Ω_n the set of offspring of n, each offspring being a decision node corresponding to a realization of the random variables $\zeta_{K_w}, \dots, \zeta_{K_w+C_w-1}$, i.e., the full retrieval order of containers in batch w. Note that n has a priori $|\Omega_n| = C_w!$ offspring.

- If n is a decision node, then $r(n)$ is well defined and is equal to the number of containers blocking the target container in configuration n (i.e., the $(C-\lambda_n+1)^{th}$ container to be retrieved). We denote by Δ_n the set of offspring of n, which can either be chance nodes if there exists $w \in \{1, \dots, W\}$ such that $\lambda_n + 1 = C - K_w + 1$ or decision nodes otherwise. For the sake of simplicity, we compute Δ_n greedily by considering all feasible combinations of relocations of the $r(n)$ containers blocking the target container in n. Note that $|\Delta_n|$ is of the order of

$(S-1)^{r(n)}$, where S is the number of stacks.

Equation (4.1) provides the relation to compute the cost-to-go by backtracking. For all n in the decision tree, we have

$$f(n) = \begin{cases} \dfrac{1}{|\Omega_n|} \displaystyle\sum_{n_i \in \Omega_n} f(n_i) \text{ , } \textit{if } n \textit{ is a chance node,} \\ r(n) + \displaystyle\min_{n_i \in \Delta_n} \{f(n_i)\} \text{ , } \textit{if } n \textit{ is a decision node.} \end{cases} \tag{4.3}$$

In the case in which the probability of each permutation (in each batch) is not uniform, we mentioned that in practice, operators provide the probability of potential orders for each batch. Given a chance node n and one of its offspring $n_i \in \Omega_n$, this input probability is exactly the probability that the actual retrieval order is the one revealed in n_i. For a given node n, we denote these probabilities by $(p_{n_i})_{n_i \in \Omega_n}$. In this case, Equation (4.3) is replaced by:

$$f(n) = \begin{cases} \displaystyle\sum_{n_i \in \Omega_n} p_{n_i} f(n_i) \text{ , } \textit{if } n \textit{ is a chance node,} \\ r(n) + \displaystyle\min_{n_i \in \Delta_n} \{f(n_i)\} \text{ , } \textit{if } n \textit{ is a decision node,} \end{cases}$$

for all n in the decision tree.

Figures 4-4 and 4-5 provide the description of the decision tree corresponding to the example in Figure 4-2, using chance/decision nodes and configurations, respectively. A chance node is depicted with a circle, and a decision node with a square.

To illustrate how to use Equation (4.3), we derive the calculations using the example in Figure 4-4. Suppose that $f(25) = f(26) = f(27) = 0.5$ are known, then we get $f(17) = f(18) = f(19) = 0.5$, $f(07) = f(09) = 1.5$, and $f(08) = 2.5$, leading to $f(01) = 3.5$. Similarly, by backtracking, we can compute $f(02) = f(03) = f(04) = 1.5$ and $f(05) = f(06) = 2.5$, giving us $f(00) = 13/6$.

As the example shows, considering the full decision tree can become intractable even for small examples, hence quickly impossible for larger problems. As previously mentioned, the number of decision offspring of a chance node scales with $C_w!$, and the number of offspring of a decision node is of the order of $(S-1)^{r(n)}$. So the size of the

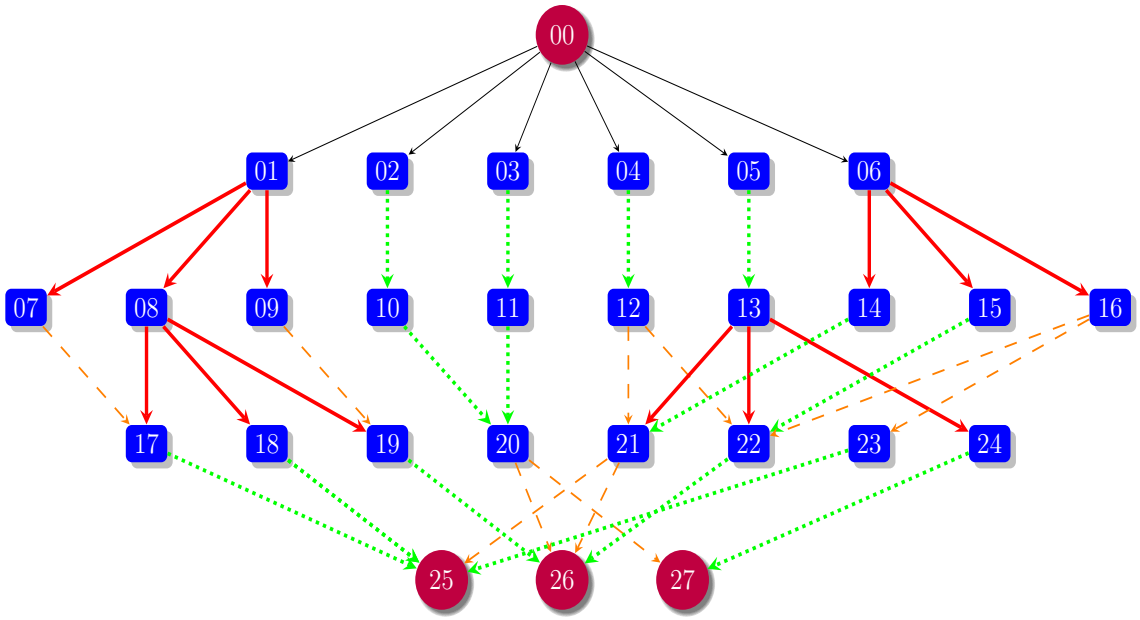


Figure 4-4: Decision tree represented with nodes. The colored arrows represent different values of immediate cost, i.e., the number of containers blocking the target container (dotted green: 0, dashed orange: 1, thick solid red: 2).

tree grows exponentially with the size of the problem. However, there exist general and specific techniques to reduce substantially the size of this tree, as we discuss now.

First, there are suboptimal approaches. One way to approximate $f(n)$, when n is a chance node, is to sample from its offspring. When Ω_n is large (which can happen with large batches), one might sample a certain number of realizations, resulting in a set of offspring $\Psi_n \subset \Omega_n$. By sampling "enough," we show in Section 4.6, we can ensure guarantees on expectation for such an algorithm. Another popular suboptimal approach is to use techniques from approximate dynamic programming. These techniques can provide good empirical results, but no guarantee on how far from optimality can be obtained. This direction is not discussed in this chapter but can be a direction for future work. Finally, another approach is to consider heuristics such as the ones described in the next section, which select a subset of the offspring of decision nodes, and lead to upper bounds on the optimal value $f(0)$.

Second, there exist ways to decrease the size of the tree while ensuring optimality. One is to reduce the number of nodes using the problem structure of the SCRP.

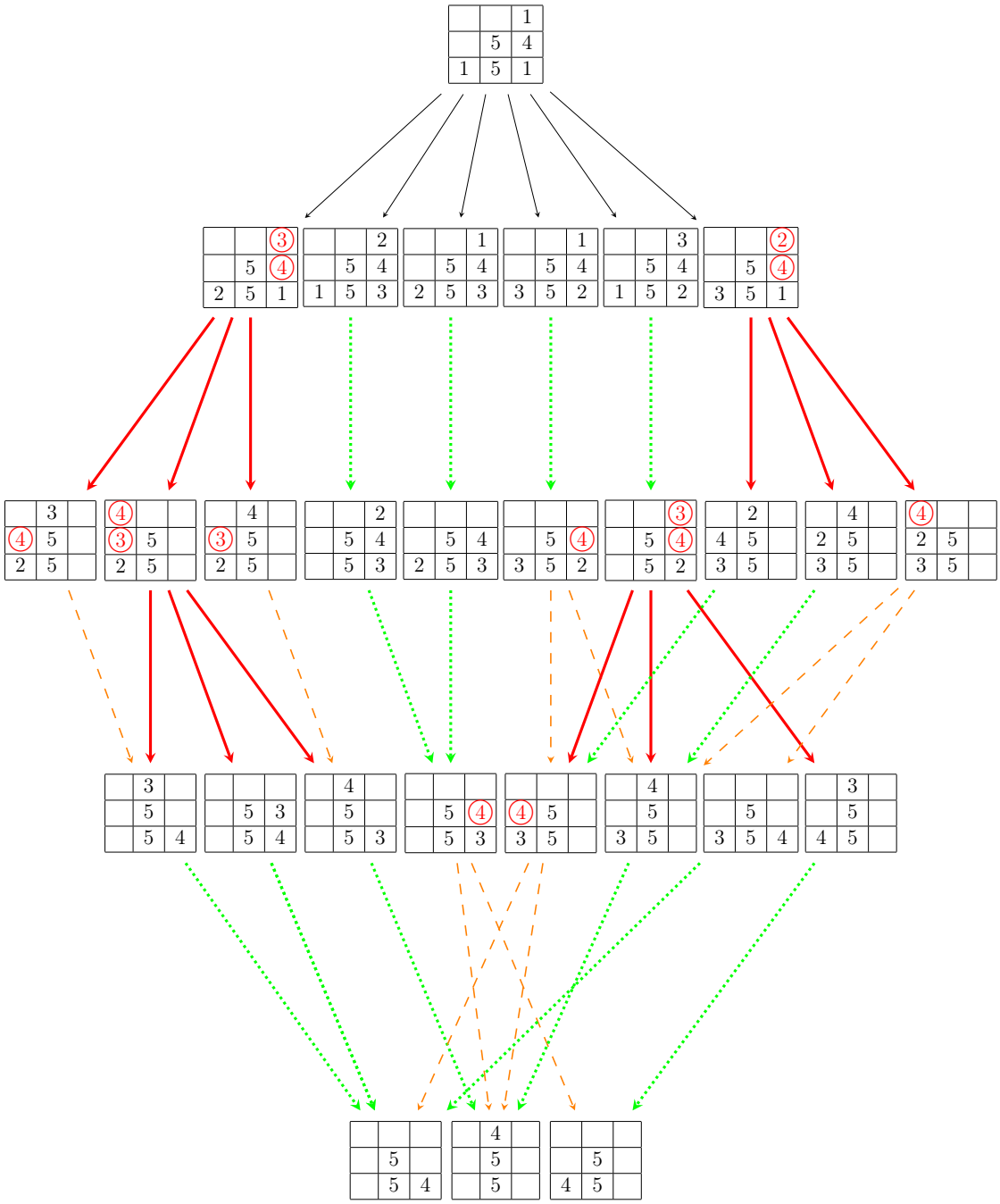


Figure 4-5: Decision tree represented with configurations. The colored arrows represent different values of immediate cost, i.e., the number of containers blocking the target container (dotted green: 0, dashed orange: 1, thick solid red: 2). Red circled numbers highlight containers blocking the target container.

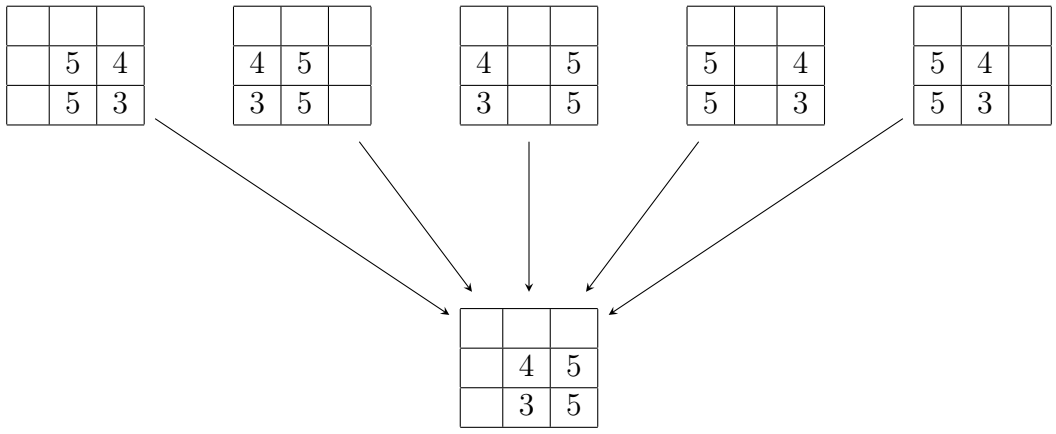


Figure 4-6: Abstraction procedure.

In the online setting, Ku and Arthanari [43] propose an *abstraction* technique, that significantly shrinks the size of the tree. Thanks to Assumption A_4^*, we can consider that the stacks of a configuration are interchangeable, making many configurations equivalent in terms of number of relocations. For instance, in Figure 4-4, nodes 20 and 21 are identical in terms of number of relocations.

We describe the abstraction step with an example in Figure 4-6. The five configurations at the top are all equivalent to the configuration at the bottom. The abstraction transformation first ranks the stacks by increasing height. For stacks with equal height, it breaks ties by ranking them lexicographically going from bottom to top. Stacks are rearranged to have the first ranked on the left and the last on the right. Ku and Arthanari [43] use a slightly different rule, and add the extra step of removing empty stacks. Using the abstraction procedure, the proposed algorithm should not generate nodes twice with identical abstracted versions. Even though Ku and Arthanari [43] introduce this rule for the online model, this abstraction step also applies in the batch setting. Throughout the rest of the chapter, we will refer in pseudocode to the function ABSTRACT(n), when applying this method to a given node n. We mention that Ku and Arthanari [43] also suggest caching strategies that could be added on the top of this simplification step, including caching part of the tree or using a transportation table.

Finally, the performance of a decision-tree-based algorithm depends on the exploration strategy of the tree. For the online model, Ku and Arthanari [43] use depth-first-search (DFS), and we propose to explore the best-first-search (BFS) approach. Note that BFS requires some kind of measure that we define in Section 4.5.

In further sections, we explore two other ways to decrease the size of the tree while still ensuring optimality. The first one is specific to the SCRP, and uses properties of the problem to increase λ^*. Recall that λ^* is the minimum level of the tree at which we can find the optimal expected number of relocations, without further branching. We show that we can set λ^* to $\max\{S, C_W\}$, where S is the number of stacks, and C_W is the number of containers in the last batch. Comparatively, Ku and Arthanari [43] branch until $\lambda^* = 0$. The second optimal pruning strategy uses lower bounds in a BFS scheme.

After we introduce the batch model for the SCRP (as well as the online model) and some preliminary concepts about decision trees, the next three sections develop approaches to solve the SCRP.

4.4 Heuristics and Lower Bounds

Before introducing the two main algorithms, we describe in this section heuristics and lower bounds for the SCRP. Indeed, *PBFS* and *PBFSA* build upon some of these bounds. In addition, these algorithms provide good intuition on how to solve the SCRP.

Let n be a given configuration with S stacks and T tiers. Recall that a Container c is a *blocking container* in n if it is stacked above at least one container which is to be removed before c. Note that all the bounds mentioned below apply both in the batch and online models.

4.4.1 Heuristics

For the sake of completeness of this chapter, we first describe three existing heuristics used in proofs and/or our computational experiments before describing two novel

heuristics.

Existing heuristics

Random. For every relocation of a blocking Container c from Stack s, the Random heuristic picks any Stack $s' \neq s$ uniformly at random among stacks that are not "full," i.e., stacks containing strictly fewer than T containers.

Leveling (L). For every relocation of a blocking Container c from Stack s, L chooses the Stack $s' \neq s$ currently containing the fewest of containers, breaking ties arbitrarily by selecting such leftmost stack.

Heuristic L is interesting for several reasons. Most important, it is an intuitive and commonly used heuristics in real operations in that it uses no more than the height of each stack. It does not require any information about batches or departure times, which means it is robust with respect to the inaccuracy of information. In addition, it is optimal for any configurations with S containers or fewer (see Section 4.5). Finally, we show strong evidence in the last computational experiment (see Section 4.7) that this policy is optimal for the SCRP under the online model with a unique batch (representing the case of no-information on the retrieval order).

Expected reshuffling index (ERI). This index-based heuristic was introduced in [43] for the online model. For every relocation of a blocking Container c from Stack s, ERI computes a score called the expected reshuffling index for each Stack $s' \neq s$ that is not full, denoted by $ERI(s', c)$. ERI chooses the Stack $s' \neq s$ with the lowest $ERI(s', c)$. In the case of a tie, the policy breaks it by selecting the highest stack among the ones minimizing $ERI(s', c)$. Further ties are arbitrarily broken by selecting the leftmost stack verifying the two previous conditions. $ERI(s', c)$ corresponds to the expected number of containers in Stack s' that depart earlier than c. Let $H_{s'}$ be the current number of containers in s'. If $H_{s'} = 0$, then $ERI(s', c) = 0$. Otherwise, let $(c_1, \ldots, c_{H_{s'}})$ be the containers in s', then $ERI(s', c) = \sum_{i=1}^{H_{s'}} \mathbb{1}\{c_i < c\} + \frac{\mathbb{1}\{c_i = c\}}{2}$.

First new heuristic: expected minmax (EM)

EM considers an idea similar to the MinMax heuristic in [10] for the CRP. Let $min(s)$ be the smallest label of a container in s ($min(s) = C + 1$, if Stack s is empty). For every relocation of a blocking Container c from Stack s, we select the stack to which we relocate c using the following rules:

[Rule 1]: If there exists $s' \neq s$ such that $min(s') > c$, let

$$M = \min_{s' \in \{1,\ldots,S\} \setminus s} \{\min(s') : \min(s') > c\}.$$

Select a stack such that $min(s') = M$, breaking ties by choosing from the highest ones, finally taking the leftmost stack if any ties remain.

[Rule 2]: If for all Stacks $s' \neq s,\ min(s') \leqslant c$, let

$$M = \max_{s' \in \{1,\ldots,S\} \setminus s} \{\min(s')\}.$$

Select a stack such that $min(s') = M$. If there are several such stacks, select those with the minimum number of containers labeled M. Further ties are again broken by taking the highest ones, and finally choosing the leftmost one arbitrarily.

Rule 1 says: if there is a stack where $\min(s)$ is greater than c (c can almost surely avoid being relocated again), then choose such a stack where $\min(s)$ is minimized, since stacks with larger minimums can be useful for larger blocking containers.

If there is no stack satisfying $\min(s) > c$ (Rule 2), then we have two cases following the same rule. On one hand, if $M = c$, then M is the maximum of the minimum labels of each stack, and c can potentially avoid being relocated again. If there are several stacks that maximize the minimum label, then by selecting the one with the fewest of containers labeled M, EM minimizes the probability of c being relocated again. On the other hand, if $M < c$, c will almost surely be relocated again, then EM chooses the stack where $\min(s)$ is maximized to delay the next unavoidable relocation

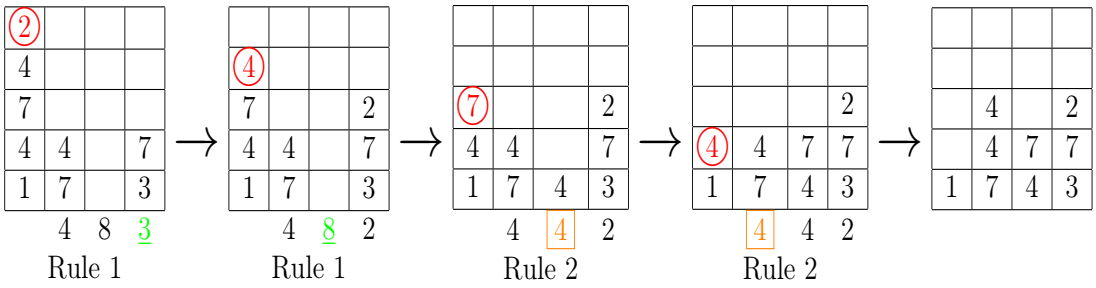


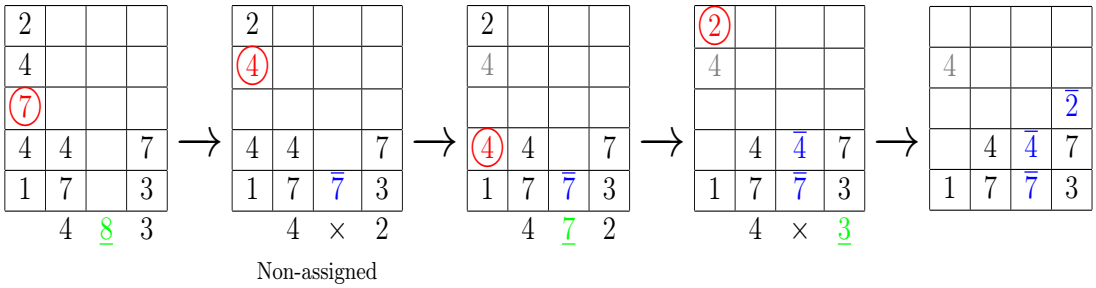
Figure 4-7: Decisions of the EM heuristic on an example with 5 tiers, 4 stacks, and 9 containers (3 per batch). Under the batch model, the first batch has been revealed and we present the decisions to retrieve the first container made by EM. The container with the circled red label is the current blocking container. Numbers under the configuration correspond to the stack indices $min(s)$. The underlined green (respectively squared orange) indices correspond to the selected stack with the corresponding M when Rule 1 (respectively Rule 2) applies.

of c as much as possible. We show how EM makes decision on a simple example in Figure 4-7.

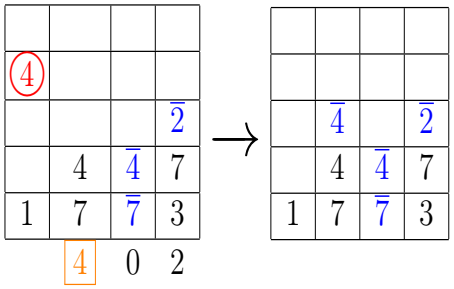
Second new heuristic: expected group assignment (EG)

EM is quite intuitive because it tries to minimize the number of blocking containers after each retrieval. EG aims for the same goal, but uses some more sophisticated rules (although, as shown in the experiments in Section 4.7, EG does not always provide better solutions than EM). EG is inspired by a heuristic designed by Wu and Ting [77] for the complete information case, and we generalize this idea to the SCRP. It is different from ERI and EM because it considers a group of blocking containers together, whereas ERI and EM consider them one at a time. EG can be decomposed in two main phases for each retrieval. The decisions made by EG on the same example as shown in Figure 4-7 are given in Figure 4-8.

The first phase assigns the blocking containers for which there exists $s' \neq s$ such that $min(s') > c$. If this is not the case, the assignments of these containers will be ignored during the first phase. The acceptable containers are assigned in descending order of labels, i.e., the container with the highest label is assigned first (breaking ties for the highest one first). To assign these acceptable containers, the first phase applies the first of the EM rules. Finally, an acceptable container cannot be assigned



4-8a. First phase: EG assigns acceptable containers in descending order. The container with the circled red label is the next acceptable container that EG tries to assign to a stack. Containers with overlined blue labels are assigned, and with gray labels are unassigned. Below, we show the indices $min(s)$ to apply the first rule of EM ($\times$ means that a container below the considered container has already been assigned to a stack).



4-8b. Second Phase, EG assigns all unassigned containers using the index $Gmin(s)$.

Figure 4-8: Decisions of the EG heuristic in an example with 5 tiers, 4 stacks, and 9 containers (3 in each batch). Under the batch model, the first batch has been revealed and we present the two-phases decisions to retrieve the first container made by EG.

to a stack if there is a container below it that was previously assigned to this stack.

The assignment in the second phase for the blocking containers not yet assigned might lead to additional relocations. These containers are assigned to other stacks in ascending order of labels. The second phase first computes a modified $min(s')$ index for each stack denoted by $Gmin(s')$, which is defined as follows: Let $H_{s'}$ be the height of Stack s' and $B(s')$ be the subset of containers assigned in the first phase to Stack

s',

$$Gmin(s') = \begin{cases} -1, \ \textit{if } |B(s')| + H_{s'} = T, \\ min(s'), \ \textit{if } |B(s')| = 0, \\ B(s'), \ \textit{if } |B(s')| = 1, \\ 0, \ \textit{otherwise.} \end{cases}$$

If a stack is full after we assign the containers during the first phase, then it cannot be selected. If no container was assigned, the index remains as the *min*. If one container was assigned, it is "artificially" the new minimum of the stack. Finally, if more than one container was assigned, the index becomes very unattractive by being as low as possible (0). The second phase is similar to the EM heuristic, but it considers *Gmin* instead of the *min* index, breaking ties identically. Note that, after each assignment in the second phase, we update *Gmin* accordingly for the remaining containers to be assigned. For more details in the complete information case, we refer the reader to [77].

4.4.2 Lower Bounds

After defining heuristics (upper bounds), we are now concerned with defining valid lower bounds for the SCRP. More specifically, we care about computing lower bounds for decision nodes in the decision tree defined before. Note that the computation of lower bounds easily extends to chance nodes.

Blocking lower bound

Suppose that the departure order is known, as in the CRP. The following lower bound was introduced in [38] and is based on the following simple observation. If a container is blocking in n, then it must be relocated at least once. Thus, the optimal number of relocations is lower bounded by the number of blocking containers.

In the SCRP, the retrieval order is a random variable, so the fact that a container is blocking is also random. Let us denote the expected number of blocking containers

in n by $b(n)$. **Note that this notation extends the notation of Chapter 3 which defines $b(.)$ in the case where the order is known**. Therefore, by taking the expectation on the retrieval order of the previous fact, which holds for every retrieval order, we have the following observation.

Observation 3. *For all configurations n, $f(n)$ is the minimum expected number of relocations to empty n, and $b(n)$ is the expected number of blocking containers, then*

$$f(n) \geqslant b(n).$$

Lemma 5 shows one way to compute the expected number of blocking containers for one stack, and $b(n)$ is the sum of the expected number of blocking containers of each stack of n. Mathematically, let $b_s(n)$ be the expected number of blocking containers in Stack s of n, we have $b(n) = \sum_{s=1}^{S} b_s(n)$.

Lemma 5. *Let n be a single stack configuration with T tiers, and $H \geqslant 0$ containers ($H \leqslant T$). If $H = 0$, we have*

$$b(n) = 0.$$

If $H \geqslant 1$, we denote the label of containers by $(c_i)_{i=1,\dots,H}$, where c_1 is the container at the bottom and c_H is the container at the top (see Figure 4-9), then we have:

$$b(n) = H - \sum_{h=1}^{H} \frac{\mathbb{1}\left\{c_h = \min\limits_{i=1,\dots,h} \{c_i\}\right\}}{\sum\limits_{i=1}^{h} \mathbb{1}\{c_h = c_i\}},$$

where $\mathbb{1}\{A\}$ is the indicator function of A.

Proof. Clearly, if $H = 0$, then $b(n) = 0$. If $H \geqslant 1$, then by definition we have

$$b(n) = \mathbb{E}\left[\sum_{h=1}^{H} \mathbb{1}\{c_h \textit{ is a blocking container}\}\right] = \sum_{h=1}^{H} \mathbb{P}\left[c_h \textit{ is a blocking container}\right].$$

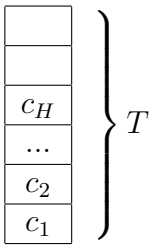


Figure 4-9: Example of a single stack configuration.

Let us fix $h \in \{1, \ldots, H\}$, and compute the probability that c_h is blocking. We consider two cases:

- If $c_h > \min_{i=1,\ldots,h} \{c_i\}$, then c_h is almost surely blocking.

- Otherwise $c_h = \min_{i=1,\ldots,h} \{c_i\}$, and there are $\sum_{i=1}^{h} \mathbb{1}\{c_h = c_i\} - 1$ containers below c_h with the same label (or batch). Since each departure sequence between containers of the same batch is equally likely, the probability that c_h is blocking is equal to $\frac{\sum_{i=1}^{h} \mathbb{1}\{c_h=c_i\}-1}{\sum_{i=1}^{h} \mathbb{1}\{c_h=c_i\}} = 1 - \frac{1}{\sum_{i=1}^{h} \mathbb{1}\{c_h=c_i\}}$.

Consequently, we get

$$\begin{aligned}
&\mathbb{P}\left[c_h \textit{ is a blocking container}\right] \\
&\quad = 1 \times \mathbb{1}\left\{c_h > \min_{i=1,\ldots,h} \{c_i\}\right\} + \left(1 - \frac{1}{\sum_{i=1}^{h} \mathbb{1}\{c_h=c_i\}}\right) \times \mathbb{1}\left\{c_h = \min_{i=1,\ldots,h} \{c_i\}\right\} \\
&\quad = 1 - \frac{\mathbb{1}\left\{c_h = \min_{i=1,\ldots,h} \{c_i\}\right\}}{\sum_{i=1}^{h} \mathbb{1}\{c_h=c_i\}}.
\end{aligned}$$

We sum the above expression for $h = 1, \ldots, H$ to conclude the proof. □

Therefore, one can compute the blocking lower bound as follows: let H^s be the number of containers in Stack s, and $(c_1^s, \ldots, c_{H^s}^s)$ be the containers in Stack s listed

from bottom to top, then

$$b(n) = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - \sum_{h=1}^{H^s} \frac{\mathbb{1}\left\{ c_h^s = \min\limits_{i=1,\dots,h} \{c_i^s\} \right\}}{\sum\limits_{i=1}^{h} \mathbb{1}\{c_h^s = c_i^s\}} \right). \tag{4.4}$$

Non-uniform case. In the case where probabilities are not uniform across retrieval orders, we still consider a similar lower bound. For each Container c_h^s, let $q_{c_h^s}$ be the probability that c_h^s is the first container to be retrieved among the ones with the same batch, and positioned below in its stack. Equation (4.4) extends to give:

$$b(n) = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - \sum_{h=1}^{H^s} q_{c_h^s} \mathbb{1}\left\{ c_h^s = \min_{i=1,\dots,h} \{c_i^s\} \right\} \right).$$

Look-ahead lower bounds

Note that the blocking lower bound b is only taking into account the current configuration. However, some relocations lead necessarily to an additional relocation. As in Chapter 3, we refer to such relocations as "bad." Formally, let s be a stack of a configuration, and $\min(s)$ be the smallest label of a container in s. Recall that, if s is empty, we set $\min(s) = C + 1$. We say that the relocation of Container c from Stack s is "bad" if $c > \max\limits_{s'=1,\dots,S,\ s' \neq s} \{\min(s')\}$. We propose to construct a lower bound that anticipates "bad" relocations.

The basic idea is based on a similar one used in [87] for the CRP. We consider the first look-ahead lower bound denoted by $b_1(n)$. By definition we take $b_1(n) = b(n) + d_1(n)$, where $b(n)$ is the blocking lower bound, and $d_1(n)$ is the expected number of unavoidable "bad" relocations while performing the first removal. We compute the term $d_1(n)$ by considering all realizations of the first target container. For each realization, we compute the number of unavoidable "bad" relocations and average them. Formally, for a given configuration n, consider U_n the set of potential next target container in n (which can be a singleton if it is known already), i.e.,

$U_n = \left\{ c \,\middle|\, c = \min_{s=1,\dots,S} (min(s)) \right\}$. Based on the definition of a bad relocation, we compute the number of unavoidable "bad" moves for each $u \in U_n$ denoted by $\beta(n, u)$, and we take:

$$d_1(n) = \frac{1}{|U_n|} \sum_{u \in U_n} \beta(n, u),$$

or $d_1(n) = \sum_{u \in U_n} p_{n,u} \beta(n, u)$, where $p_{n,u}$ is the probability that u is the next target container in n if the probabilities considered are not uniform (which can be computed using $(p_{n_i})_{n_i \in \Omega_n}$ if n is a chance node).

		4
		3
1	3	1

Figure 4-10: Example for look-ahead lower bounds.

For example, in Figure 4-10, the presented configuration denoted by n is such that $b(n) = 2$. Now consider a container $u \in U_n$: if u is the container labeled 1 in Stack 1, then there is no blocking container, so $\beta(n, u) = 0$; if u is the other container labeled 1, the relocation of the container labeled 4 from Stack 3 is necessarily a bad relocation, since $min(1) = 1 < 4$ and $min(2) = 3 < 4$, but it is not the case for the blocking container labeled by three, hence $\beta(n, u) = 1$. Therefore, $d_1(1) = 0.5(0 + 1) = 0.5$, and $b_1(n) = 2 + 0.5 = 2.5$, hence giving a lower bound closer to the optimal solution than $b(n)$. Note that, if n has an empty stack, then $\beta(n, u) = 0$ for all $u \in U_n$, hence $d_1(n) = 0$.

We can refine this idea by trying to find unavoidable "bad" relocations for the second removal. In this case, the configuration depends on the first removal and the decisions that have been made accordingly. For the sake of clarity, consider that the first target container has been revealed, and denote it u_1. After retrieving u_1, only containers blocking u_1 have changed from their initial position. It can be very challenging to detect future unavoidable "bad" moves for these containers. To bypass this issue, we consider that all containers blocking u_1 are also removed, resulting in

a configuration without u_1 and its blocking containers. Given this new configuration denoted by $n(u_1)$, we can compute the expected number of unavoidable bad moves $d_1(n(u_1))$. Since u_1 is actually random, we have to consider each scenario with its associated probability and compute a new configuration where blocking containers are retrieved with the target container. We denote the result $d_2(n)$, and it is a lower bound on the expected number of unavoidable bad relocations for the first two removals starting at n. Finally, our second look-ahead lower bound is given by $b_2(n) = b(n) + d_2(n)$.

Algorithm 1 Lower bound on number of unavoidable bad relocations for k first removals.

procedure $[d_k(n)]$ = UnavoidableBadReloc $(n,\ k)$
  if $k = 0$ or n has an empty stack or n is empty **then** $d_k(n) = 0$
  else let $U_n = \{$*containers with minimum label in* $n\}$
    if $k = 1$ **then** $d_k(n) = \frac{1}{|U_n|} \sum_{u \in U_n} \beta(n, u)$
    else
      for $u \in U_n$ **do**
        Let $n(u)$ be the configuration n without u and all containers blocking u
        Compute recursively $d_{k-1}\,(n(u))$ = UnavoidableBadReloc $(n(u),\ k-1)$
      Compute $d_k(n) = \frac{1}{|U_n|} \sum_{u \in U_n} \beta(n, u) + d_{k-1}\,(n(u))$

This idea can easily be generalized for $k \geqslant 2$ by induction with $b_k(n) = b(n) + d_k(n)$. Here k is the number of removals that the lower bound considers to compute the expected number of unavoidable bad relocations (see pseudocode of Algorithm 1). We mention that we only use the first and second look ahead lower bounds in our computational experiments. However, note that, as k grows, the computational complexity clearly increases, whereas experiments reveal that the marginal increase of the lower bound, i.e., $b_{k+1}(n) - b_k(n) \geqslant 0$, decreases.

4.5 *PBFS*, a New Optimal Algorithm for the SCRP

Building upon lower bounds introduced in the previous section, this section introduces, studies, and proves the optimality of one of the main contributions of this chapter, the *PBFS* Algorithm.

4.5.1 *PBFS* Algorithm

We start by giving the pseudocode of our algorithm, and we derive its optimality in Lemmas 6 and 7. *PBFS* takes two inputs, the configuration n for which we aim to compute $f(n)$, and a valid lower bound l. This algorithm uses a combination of four features to return $f(n)$. The first one is the BFS exploration of the tree based on a given lower bound l. We first compute f for the "most promising nodes," because nodes with small lower bounds are more likely to result in small f. The second technique is stopping to compute f recursively after level $\lambda^* = \max\{S, C_W\}$, by calculating it either using b or the A^* algorithm defined later. In Algorithm 2, $A^*(n)$ denotes the optimal number of relocations for node n obtained using the A^* algorithm. The third feature is pruning with a lower bound, revealing the sub-optimality of some nodes without actually computing f. As its fourth feature, the algorithm also uses the abstraction technique described previously.

Algorithm 2 *PBFS* Algorithm

procedure $[f(n)] = PBFS\,(n,\ l)$
  if $\lambda_n \leqslant S$ (n has fewer than S containers) **then** $f(n) = b(n)$
  else
    if n is a *chance node* **then** start with $\Psi_n^{PBFS} = \{\}$
      for $n_i \in \Omega_n$ **do** $n_i \leftarrow$ ABSTRACT(n_i)
        if there exists $m = n_i$ already in Ψ_n^{PBFS} **then** $p_m^n \leftarrow p_m^n + \frac{1}{|\Omega_n|}$
        else if there exists $m = n_i$ in decision tree **then** add m to Ψ_n^{PBFS} and $p_m^n = \frac{1}{|\Omega_n|}$
        else add n_i to Ψ_n^{PBFS}, $p_{n_i}^n = \frac{1}{|\Omega_n|}$ and compute $f(n_i) = PBFS\,(n_i,\ l)$
      Compute $f(n) = \sum_{n_i \in \Psi_n^{PBFS}} p_{n_i}^n f(n_i)$
    else n is a *decision node*
      if $\lambda_n = C_W$ (the full retrieval order is known) **then** $f(n) = A^*(n)$
      else construct Δ_n from all feasible sets of decisions to deliver the target container
        Compute $l(n_i)$ for each $n_i \in \Delta_n$
        Sort $\left(n_{(1)}, n_{(2)}, \ldots, n_{(|\Delta_n|)}\right)$ in nondecreasing order of $l(.)$
        Compute $f(n_{(1)}) = PBFS\left(n_{(1)},\ l\right)$
        Start with $\Gamma_n^{PBFS} = \{n_{(1)}\}$ and $k = 2$
        while $k \leqslant |\Delta_n|$ and $l(n_{(k)}) < \min_{j=1,\ldots,k-1} \left\{f(n_{(j)})\right\}$ **do** $n_{(k)} \leftarrow$ ABSTRACT$(n_{(k)})$
          if there exists $m = n_{(k)}$ already in the decision tree **then** add m to Γ_n^{PBFS}
          else add $n_{(k)}$ to Γ_n^{PBFS} and compute $f(n_{(k)}) = PBFS\left(n_{(k)},\ l\right)$
          Update $k = k + 1$
        $f(n) = r(n) + \min_{n_i \in \Gamma_n^{PBFS}} \{f(n_i)\}$

Decreasing the size of decision tree by increasing λ^* to $\max\{S, C_W\}$

If $C_W \leqslant S$, then compute $\mathbf{f}(\mathbf{n})$ using $\mathbf{b}(\mathbf{n})$. Recall that, for every relocation, heuristic L chooses the stack with the fewest of containers, breaking ties arbitrarily by choosing the leftmost one. Note that L always provides a valid upper bound for the SCRP. So if we denote the resulting expected number of relocations to empty configuration n using L by $f_L(n)$, then we have $f_L(n) \geqslant f(n)$.

Lemma 6. *Let n be a configuration with S stacks, T tiers, and C containers such that $C \leqslant S$, then we have*

$$f_L(n) = b(n) = f(n)$$

Proof. Consider a retrieval order of containers from n that has a nonzero probability of occurring. We now apply and explain Observation 2: If there are no blocking containers, then the lemma clearly holds. Otherwise, let c be one of the blocking containers for this retrieval order, and consider the first removal for which c has to be relocated. For this removal, there are at most S containers in the configuration, hence there exists at least one empty stack to relocate c. Since heuristic L chooses always empty stacks if one exists, L would move c to one of the existing empty stacks. Note that in this case, c will never be blocking again, hence will never be relocated again. This observation holds for any blocking containers, thus L relocates each blocking container at most once.

Since this fact holds for any retrieval orders with nonzero probability, by taking expectation on the retrieval order, we have $f_L(n) \leqslant b(n)$, thus $f_L(n) \leqslant b(n) \leqslant f(n) \leqslant f_L(n)$, which concludes the proof. □

Lemma 6 states that for configurations with S containers or fewer, heuristic L is optimal for the SCRP. This implies that, for nodes at level S, we have access to the cost-to-go function using $b(n)$, as well as an optimal solution (provided by heuristic L). Hence *PBFS* can stop branching at $\lambda^* = S$ (line 2 of Algorithm 2).

If $C_W > S$, then compute f(n) using the A^* algorithm. If n is a decision node at level C_W, the full order of retrieval is known, and computing $f(n)$ reduces to solving a classical CRP, so we can leverage the existence of efficient solutions to the classical CRP such as the A^* algorithm, and take $\lambda^* = C_W$. Throughout the rest of the chapter, A^* refers to the improved version of this algorithm presented in [4] and we denote the optimal number of relocations obtained by $A^*(n)$ (line 11 of Algorithm 2).

Combining with the two previous observations, we can take $\lambda^* = \max\{S, C_W\}$.

Decreasing the size of decision tree by pruning using lower bounds

We would also like to reduce the size of the tree before level λ^*. For a decision node n, *PBFS* considers only a subset Γ_n^{PBFS} of all the offspring Δ_n (line 21 of Algorithm 2). Our goal is to set Γ_n^{PBFS} to still guarantee optimality.

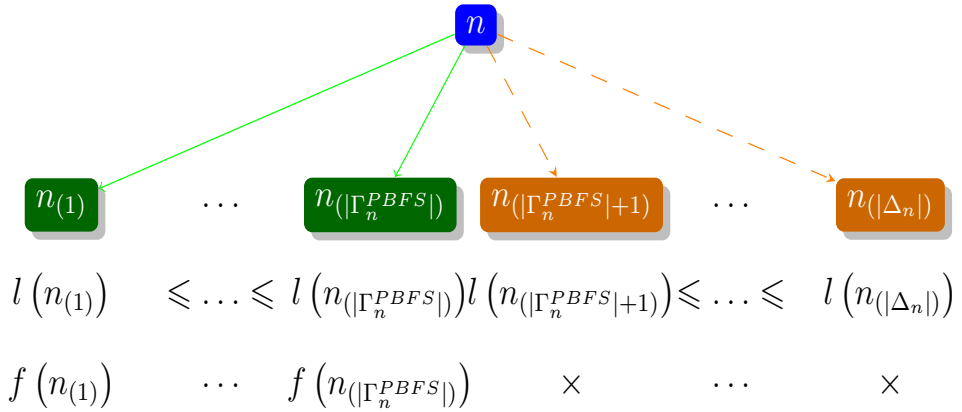


Figure 4-11: Illustration of the pruning rule. First, offspring are ordered by nondecreasing lower bounds. Then we start computing the objective function starting at $n_{(1)}$. We stop computing the objective functions once the pruning rule is reached. In the figure above, green nodes linked with full green arrows are nodes in Γ_n^{PBFS}, i.e., $f(.)$ has been computed. Orange nodes linked with dashed orange arrows are nodes in $\Delta_n \setminus \Gamma_n^{PBFS}$, i.e., $f(.)$ does not need to be computed which is represented here by $\times$.

First, *PBFS* generates all nodes $n_i \in \Delta_n$ by considering all feasible sets of decisions to deliver the target container in n (line 12 of Algorithm 2), and for each of them,

compute a lower bound $l(n_i)$, where l is the input lower bound (line 13 of Algorithm 2). Let $\left(n_{(1)}, n_{(2)}, \ldots, n_{(|\Omega_n|)}\right)$ be the list of offspring of n sorted by nondecreasing lower bound (line 14 of Algorithm 2). The algorithm considers first $n_{(1)}$, adds it to Γ_n^{PBFS} and computes $f(n_{(1)})$ recursively (lines 15-16 of Algorithm 2). Then for $k \geqslant 2$, we consider $n_{(k)}$'s sequentially and check if $l(n_{(k)}) < \min\limits_{j=1,\ldots,k-1} \left\{f(n_{(j)})\right\}$ (line 17 of Algorithm 2). If so, add $n_{(k)}$ to Γ_n^{PBFS} and compute $f(n_{(k)})$ recursively. If not, we stop branching on all nodes $n_{(k)}, \ldots, n_{(|\Omega_n|)}$. An illustration of the pruning rule is shown in Figure 4-11 and the next lemma shows the optimality of this rule.

Lemma 7. *Let n be a decision node in the decision tree, and Γ_n^{PBFS} be the subset of nodes considered for this node in Algorithm 2, and constructed as aforementioned, then we have*

$$\min_{m_i \in \Gamma_n^{PBFS}} \{f(m_i)\} = \min_{n_i \in \Delta_n} \{f(n_i)\}.$$

Proof. Let $\left(n_{(1)}, n_{(2)}, \ldots, n_{(|\Omega_n|)}\right)$ be the list of offspring of n, sorted by nondecreasing lower bounds. We consider two cases.

- If $\Gamma_n^{PBFS} = \Delta_n$, the statement clearly holds.
- Otherwise, there exists $k \leqslant |\Delta_n|$ such that $l(n_{(k)}) \geqslant \min\limits_{j=1,\ldots,k-1} \left\{f(n_{(j)})\right\}$, and $\Gamma_n^{PBFS} = \left\{n_{(1)}, \ldots, n_{(k-1)}\right\}$. Note that we have $\forall k' \geqslant k$, $f(n_{(k')}) \geqslant l(n_{(k')}) \geqslant l(n_{(k)}) \geqslant \min\limits_{j=1,\ldots,k-1} \left\{f(n_{(j)})\right\}$. Hence

$$\min_{n_i \in \Delta_n} \{f(n_i)\} = \min_{j=1,\ldots,k-1} \left\{f(n_{(j)})\right\} = \min_{m_i \in \Gamma_n^{PBFS}} \{f(m_i)\}.$$

□

We claim that increasing λ^* to $\max\{S, C_W\}$ together with pruning in a Best-First-Search scheme, dramatically helps the efficiency of *PBFS* while guaranteeing optimality. In the case of small batches, the *PBFS* algorithm appears to be efficient (see Section 4.7). However, this algorithm faces the issue that $|\Omega_n| = C_w!$ if n is a chance node. So if $\mathbf{C_w}$ **is large, typically** $\mathbf{C_w \geqslant 4}$, the number of nodes to consider

gets too large. We tackle this issue by considering a near-optimal algorithm in the next section.

As a final remark, we note that batches should be as small as possible if only information is at stake. Indeed, smaller batches correspond to an efficient information system since more information is known about the retrieval order. But the size of batches is restricted by two intrinsic constraints:

1. **Batches should be at least larger than a certain size.** Indeed, a terminal offers time slots for trucks to register, and these slots cannot be too brief, as trucks would most certainly not arrive during their appointed slot due to traffic or other uncertain factors. Therefore, given the minimum time of a slot, the terminal will allow at least a certain number of trucks to register for each slot, i.e., the minimum batch size.

2. **Batches cannot be too large** for the batch model to be applicable, since in this model, the appointment time windows are supposed to be the same as or shorter than the target waiting time. As the target waiting time is limited, only a limited number of containers can be retrieved in a certain batch.

This leads us to consider an alternative to *PBFS* (see Section 4.6) in the case of larger batches.

4.6 *PBFSA*, Near-Optimal Algorithm with Guarantees for Large Batches

Building upon *PBFS* introduced in the previous section, this section describes the randomized algorithm *PBFSA* and shows some theoretical guarantees on expectation. This new algorithm is identical to *PBFS* except when computing the value function of a chance node (lines 4 to 17 of Algorithm 3). To decrease the number of decision offspring to consider for each chance node, we sample a certain number of *i.i.d.* permutations and only consider the decision nodes associated with these permutations

as illustrated in Figure 4-12. In particular, *PBFSA* uses $f_{min}(.)$ and $f_{max}(.)$, lower and upper bound functions on $f(.)$ for offspring of chance nodes. In this paper, we use certain $f_{min}(.)$ and $f_{max}(.)$ defined in Equations (4.9)-(4.10), although others could be used. Combined with the fact that our problem has a finite number of sampling stages, this allows us to independently sample nodes to approximate the objective function. Using concentration inequalities, we can choose the number of samples needed to control the approximation error.

Algorithm 3 *PBFSA* Algorithm

procedure $[\tilde{f}(n)] = PBFSA\,(n,\ l,\ \epsilon)$
  if $\lambda_n \leqslant S$ **then** $\tilde{f}(n) = b(n)$
  else
    if n is a *chance node* **then** $\Psi_n^{PBFSA} = \{\}$. Let w_{min} be such that $\lambda_n = C - K_{w_{min}} + 1$
      Compute $\delta_n = \min\left\{w \in \{w_{min}, \ldots, W\} \,\middle|\, \sum_{u=w_{min}}^{w} C_u \geqslant \lambda_n - \lambda^*\right\}$ to get $\epsilon_n = \frac{\epsilon}{\delta_n}$
      Set $f_{max}(n)$, $f_{min}(n)$ from (4.9)-(4.10) to get $N_n(\epsilon_n) = \frac{\pi\left(f_{max}(n) - f_{min}(n)\right)^2}{2\epsilon_n^2}$
      if $N_n(\epsilon_n) \leqslant C_{w_{min}}!$ **then**
        for $i = 1, \ldots, N_n(\epsilon_n)$ **do**
          Sample a random permutation to get $n_i \in \Omega_n$ and $n_i \leftarrow$ ABSTRACT(n_i)
          if there is $m = n_i$ already in Ψ_n^{PBFSA} **then** $p_m^n \leftarrow p_m^n + \frac{1}{N_n(\epsilon_n)}$
          else if there is $m = n_i$ in decision tree **then** add m to Ψ_n^{PBFSA}, $p_m^n = \frac{1}{N_n(\epsilon_n)}$
          else add n_i to Ψ_n^{PBFSA}, $p_{n_i}^n = \frac{1}{N_n(\epsilon_n)}$ and $\tilde{f}(n_i) = PBFSA\,(n_i,\ l,\ \epsilon - \epsilon_n)$
      else
        for $n_i \in \Omega_n$ **do** $n_i \leftarrow$ ABSTRACT(n_i)
          if there exists $m = n_i$ already in Ψ_n^{PBFSA} **then** $p_m^n \leftarrow p_m^n + \frac{1}{|\Omega_n|}$
          else if there is $m = n_i$ in decision tree **then** add m to Ψ_n^{PBFSA} and $p_m^n = \frac{1}{|\Omega_n|}$
          else add n_i to Ψ_n^{PBFSA}, $p_{n_i}^n = \frac{1}{|\Omega_n|}$ and $\tilde{f}(n_i) = PBFSA\,(n_i,\ l,\ \epsilon - \epsilon_n)$
      Compute $\tilde{f}(n) = \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \tilde{f}(n_i)$
  else n is a *decision node*
    if $\lambda_n \leqslant C_W$ **then** $\tilde{f}(n) = A^*(n)$
    else Construct Δ_n from all feasible sets of decisions to deliver the target container
      Compute $l(n_i)$ for each $n_i \in \Delta_n$
      Sort $\left(n_{(1)}, n_{(2)}, \ldots, n_{(|\Delta_n|)}\right)$ in nondecreasing order of $l(.)$
      Compute $\tilde{f}(n_{(1)}) = PBFSA\left(n_{(1)},\ l,\ \epsilon\right)$
      Start with $\Gamma_n^{PBFSA} = \{n_{(1)}\}$ and $k = 2$
      while $k \leqslant |\Delta_n|$ and $l(n_{(k)}) < \min_{j=1,\ldots,k-1}\left\{\tilde{f}(n_{(j)})\right\}$ **do** $n_{(k)} \leftarrow$ ABSTRACT$(n_{(k)})$
        if there exists $m = n_{(k)}$ already in the decision tree **then** add m to Γ_n^{PBFSA}
        else add $n_{(k)}$ to Γ_n^{PBFSA} and compute $\tilde{f}(n_{(k)}) = PBFSA\left(n_{(k)},\ l,\ \epsilon\right)$
        Update $k = k + 1$
      $\tilde{f}(n) = r(n) + \min_{n_i \in \Gamma_n^{PBFSA}}\left\{\tilde{f}(n_i)\right\}$

Since we perform a sampling at each chance node, *PBFSA* incurs an approximation error at each chance node. Lemma 8 proves that the total approximation error at the root node is on average the sum of all approximation errors from the root node to any leaf node. Therefore, consider a node n, then δ_n is the number of chance nodes between n and any leaf node in the decision tree. If the target error is ϵ at node n, *PBFSA* "allocates" evenly the remaining error to the next chance nodes, giving $\epsilon_n = \frac{\epsilon}{\delta_n}$ error at each remaining chance node where sampling occurs.

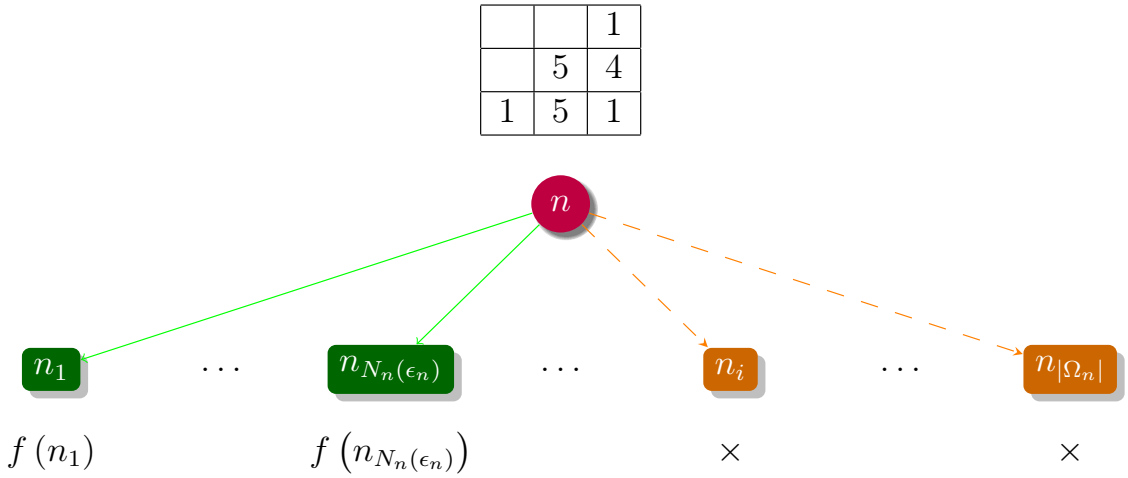


Figure 4-12: Illustration of the sampling rule. In this figure, the smallest batch is batch 1; therefore $w_{min} = 1$, and there are 6 containers, thus $\lambda_n = 6$. We have $\lambda^* = 3$ so $\delta_n = 1$, and thus $\epsilon_n = \epsilon$. These values allow us to compute the number of samples required $N_n(\epsilon_n)$. If $N_n(\epsilon_n)$ in less than the total number of offspring $|\Omega_n| = C_{w_{min}}! = 3!$, then we only compute $f(.)$ for sampled nodes. Ψ_n^{PBFSA} represents the subset of sampled nodes colored green and linked with full green arrows, and for which $f(.)$ needs to be computed. Note that $\left|\Psi_n^{PBFSA}\right| = N_n(\epsilon_n)$. Orange nodes linked with dashed orange arrows are nodes in $\Omega_n \setminus \Psi_n^{PBFSA}$, i.e., there were not sampled and $f(.)$ does not need to be computed which is represented here by $\times$. Finally, the approximate value of $f(n)$ is the average of the objective values over all sampled nodes.

Formally, let n be a given chance node, recall that *PBFS* computes $f(n) = \frac{1}{|\Omega_n|} \sum_{n_i \in \Omega_n} f(n_i)$, where each $n_i \in \Omega_n$ represents one retrieval order (a random permutation) of batch w (if $\lambda_n = C - K_w + 1$). Instead, *PBFSA* computes the number of chance nodes between n and a leaf node denoted by δ_n. If ϵ is the target error at node n, the algorithm allocates $\epsilon_n = \frac{\epsilon}{\delta_n}$ error for sampling at node n and the remaining

(i.e., $\epsilon - \epsilon_n$) to its offspring (lines 12 and 17 of Algorithm 3). Using ϵ_n, we compute $N_n(\epsilon_n)$ and obtain a subset $\Psi_n \subset \Omega_n$ from $N_n(\epsilon_n)$ offspring drawn *i.i.d.* uniformly from Ω_n. The important part is to define $N_n(\epsilon_n)$, such that $\tilde{f}(n) = \frac{1}{|\Psi_n|} \sum_{m \in \Psi_n} f(m)$ is a "good" approximation of $f(n)$, i.e., $\left|\tilde{f}(n) - f(n)\right|$ is bounded by ϵ on average. Note that if $N_n(\epsilon_n) > C_{w_{min}}!$, we would need to sample more elements than the total number of offspring of n, and thus we do not sample (lines 14-17 of Algorithm 3).

PBFSA takes three input arguments, the configuration n for which we want to evaluate f, a valid lower bound l and an upper bound $\epsilon > 0$ on the total expected "error" ensured by the algorithm. It outputs $\tilde{f}(n)$, which is a randomized approximation of $f(n)$. Because of the samplings performed in line 9 in Algorithm 3, the output of *PBFSA* is random. The average error incurred by the algorithm is $\mathbb{E}\left[\left|\tilde{f}(n) - f(n)\right|\right]$, where the expectation is taken over the aforementioned samplings. Our main result (Lemma 8) states that *PBFSA* ensures $\mathbb{E}\left[\left|\tilde{f}(n) - f(n)\right|\right] \leqslant \epsilon$, in other words, *PBFSA* guarantees an average error of at most ϵ. The proof of Lemma 8 can be found in Appendix.

Lemma 8. *Let n be a configuration with $\lambda_n \geqslant 0$ containers, l be a valid lower bound function, and $\epsilon > 0$. If $\tilde{f}(n) = PBFSA(n, l, \epsilon)$, then*

$$\mathbb{E}\left[\left|\tilde{f}(n) - f(n)\right|\right] \leqslant \epsilon.$$

Sketch of the proof. Lemma 8 is proven by backtracking from leaf nodes to the root node, i.e., consider a node n at level λ_n, then the proof is done by induction on λ_n. To show that the expected absolute value of the error at node n (i.e., $\tilde{f}(n) - f(n)$) is bounded by ϵ, we actually show that the expected positive and negative parts of the error are both bounded by $\epsilon/2$, which implies our result.

First, we consider the case where n is a decision node and we show that there is no additional error incurred by *PBFSA* at such node, i.e., if all the offspring of node n (in Γ_n^{PBFSA} and at level $\lambda_n - 1$) have the expected positive and negative parts of their error bounded by $\epsilon/2$ (the induction hypothesis), then the expected positive and negative parts of the error at node n are also bounded by $\epsilon/2$.

Second, we consider the case where n is a chance node, and we show that an additional error is incurred due to sampling. Using the lemmas proven below, the positive and negative parts of this additional error are bounded by $\epsilon_n/2$. Since all the offspring of node n (in Ψ_n^{PBFSA}) are decision nodes at level λ_n and *PBFSA* sets a target error of $(\epsilon - \epsilon_n)/2$ for these nodes, then the first part of the proof shows that the positive and negative parts of the error of each offspring of node n are bounded by $(\epsilon - \epsilon_n)/2$. Combining both observations leads to n having the positive and negative parts of its error bounded by $\epsilon/2$, which proves the lemma. □

4.6.1 Hoeffding's Inequality Applied to the SCRP

To prove this result, we use Hoeffding's inequality to compute the number of samples to ensure probabilistic guarantees. We first state the well-known inequality and a direct corollary.

Theorem 2 (Hoeffding's inequality)**.** *Let $X \in [x_{min}, x_{max}]$ be a real-valued bounded random variable with mean value $\mathbb{E}[X]$. Let $N \in \mathbb{N}$ and $(X_1, \ldots, X_N)$ be N i.i.d. samples of X. If $\overline{X} = \dfrac{1}{N}\sum_{i=1}^{N} X_i$, then we have*

$$\forall \delta > 0 \ , \ \mathbb{P}\left(\overline{X} - \mathbb{E}[X] > \delta\right) \ \leqslant \ exp\left(\frac{-2N\delta^2}{(x_{max} - x_{min})^2}\right), \tag{4.5}$$

and

$$\forall \delta > 0 \ , \ \mathbb{P}\left(\overline{X} - \mathbb{E}[X] < -\delta\right) \ \leqslant \ exp\left(\frac{-2N\delta^2}{(x_{max} - x_{min})^2}\right). \tag{4.6}$$

Corollary 2. *Let $X \in [x_{min}, x_{max}]$ be a real-valued bounded random variable with mean value $\mathbb{E}[X]$. Let $N \in \mathbb{N}$ and $(X_1, \ldots, X_N)$ be N i.i.d. samples of X. If $\overline{X} = \dfrac{1}{N}\sum_{i=1}^{N} X_i$, then $\forall \epsilon > 0$ such that $N \geqslant \dfrac{\pi\,(x_{max} - x_{min})^2}{2\epsilon^2}$, we have*

$$\mathbb{E}\left[\left(\overline{X} - \mathbb{E}[X]\right)^+\right] \leqslant \frac{\epsilon}{2}, \tag{4.7}$$

$$\mathbb{E}\left[\left(\overline{X} - \mathbb{E}[X]\right)^-\right] \leqslant \frac{\epsilon}{2}, \tag{4.8}$$

where $x^+ = \max\{x, 0\}$ *(resp.* $x^- = -\min\{x, 0\}$*) is the positive (resp. negative) part of* x*.*

To use Hoeffding's inequality, we need to define lower (f_{min}) and upper (f_{max}) bound functions, such that for each chance node n, $f_{min}(n) \leqslant \min_{n_i \in \Omega_n} \{f(n_i)\}$ and $f_{max}(n) \geqslant \max_{n_i \in \Omega_n} \{f(n_i)\}$.

Lemma 9. *Let* n *be a chance node, if*

$$f_{min}(n) = \min_{n_i \in \Omega_n} \{b(n_i)\}, \tag{4.9}$$

and

$$f_{max}(n) = \min \left\{ \left((\lambda_n - S)(T-1)\right)^+ + \left(\min\{S, \lambda_n\} - 1\right) \ , \ \left(2\left\lceil \frac{\lambda_n}{S} \right\rceil - 1\right) \max_{n_i \in \Omega_n} \{b(n_i)\} \right\}, \tag{4.10}$$

then

$$f_{min}(n) \leqslant \min_{n_i \in \Omega_n} \{f(n_i)\} \ \textit{and} \ f_{max}(n) \geqslant \max_{n_i \in \Omega_n} \{f(n_i)\}.$$

Notice that the previous lemma involves computing $\min_{n_i \in \Omega_n} \{b(n_i)\}$ and $\max_{n_i \in \Omega_n} \{b(n_i)\}$. The following corollary provides an efficient way to compute these values.

Lemma 10. *Let* n *be a chance node, and* $w_{min} \in \{1, \dots, W\}$ *be such that* $\lambda_n = C - K_{w_{min}} + 1$ *(i.e., the minimum batch in* n*). For each Stack* s *of* n *with* $H^s \geqslant 1$ *containers, let* $(c_h^s)_{h=1,\dots,H^s}$ *be the containers in* s*, where* c_1^s *is the container at the bottom and* $c_{H^s}^s$ *at the top (see Figure 4-9, for the case* $H = H^s$*). Finally, consider* $C_{w_{min}}^s = |\{c_h^s = K_{w_{min}}, \ h = 1, \dots, H^s\}|$*. Then we have*

$$\min_{n_i \in \Omega_n} \{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - C_{w_{min}}^s - \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} \frac{\mathbb{1}\left\{c_h^s = \min_{i=1,\dots,h} \{c_i^s\}\right\}}{\sum_{i=1}^{h} \mathbb{1}\{c_h^s = c_i^s\}} \right), \tag{4.11}$$

and

$$\max_{n_i \in \Omega_n} \{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} \frac{\mathbb{1}\left\{ c_h^s = \min\limits_{i=1,\dots,h} \{c_i^s\} \right\}}{\sum\limits_{i=1}^{h} \mathbb{1}\{c_h^s = c_i^s\}} \right). \tag{4.12}$$

Non-uniform case. Similar to the blocking lower bound, we can extend Lemma 10 to the case where probabilities are not uniform across retrieval orders. Recall that $q_{c_h^s}$ denotes the probability that c_h^s is the first one to be retrieved among the ones positioned below in its stack and with the same label. Then we have

$$\min_{n_i \in \Omega_n} \{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - C^s_{w_{min}} - \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} q_{c_h^s} \mathbb{1}\left\{ c_h^s = \min_{i=1,\dots,h} \{c_i^s\} \right\} \right),$$

and

$$\max_{n_i \in \Omega_n} \{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} q_{c_h^s} \mathbb{1}\left\{ c_h^s = \min_{i=1,\dots,h} \{c_i^s\} \right\} \right).$$

4.7 Computational Experiments

Having introduced lower and upper bounds, *PBFS*, *PBFSA*, and theoretical guarantees in previous sections, we present several experimental results in this section to understand the effectiveness of our algorithms for the SCRP. For clarity, we refer to the set of instances from [43] as the *existing dataset*. We present four sets of experiments:

1. Based on instances from the existing dataset, which have relatively small batches, we test the *PBFS* algorithm, as well as the two new heuristics and our lower bounds.

2. We slightly modify the existing dataset to obtain the *modified dataset*, to obtain instances with relatively larger batches. We test the efficiency of *PBFSA* on this

modified dataset.

3. Based on the existing dataset, we show that *PBFS* improves on the algorithm proposed in [43] for the online model. Moreover, the two new heuristics (EM and EG) outperform the ERI algorithm on expectation for the majority of the instances in the dataset.

4. We change the existing dataset by considering that all containers belong to a unique batch. We show strong computational evidence to support Conjecture 1, which states that the leveling policy is optimal for the SCRP under the online model with a unique batch.

All experiments are performed on a MacBook Pro with 2.2 GHz Intel Core i7 processor and 8.00 GB of RAM, and the programming language is MATLAB 2016a. Finally, all results and instances used in this section are available at https://github.com/vgalle/StochasticCRP.

Implementation of heuristics.

1. Computing the number of relocations using b when S or fewer containers remain: In the retrieval process, **when there are S containers or fewer remaining in the configuration, the expected number of relocations performed by ERI, EM,EG, and L is computed using b**. This is motivated by the following observation: ERI, EM, EG, and L are optimal when S or fewer containers remain in the configuration, and Lemma 6 shows that the optimal expected number of relocations in this case is equal to b. Therefore, for all heuristics (except Random), instead of running simulations until there are no containers left, we stop when there are S containers left and compute the expected number of relocations using b instead.

2. Estimating the expected number of relocations using sampling: To estimate the exact objective value for a given heuristic, one would have to consider all possible retrieval scenarios. Instead, for each heuristic **unless specified otherwise,**

we report the average over 5000 samples (of retrieval orders) for each instance, where samples are uniformly drawn at random.

Existing dataset description. The full description of the dataset can be found in [43] and the original dataset is available at http://crp-timewindow.blogspot.com. Note that:

- Configuration sizes vary from $\mathbf{T = 3, \ldots, 6}$ tiers, and $\mathbf{S = 5, \ldots, 10}$ stacks.
- Two occupancy rates are considered, **50 and 67 percent**. The occupancy rate ($\mu \in [0, 1]$) is defined such that the initial number of containers is $C = round\,(\mu \times S \times T)$, where $round(x)$ rounds x to the closest integer. Therefore, a given triplet (T, S, μ) is equivalent to a given triplet (T, S, C), and note that if $C = round\,(0.67 \times S \times T)$, the condition $0 \leqslant C \leqslant ST - (T - 1)$ is satisfied.
- Given a configuration size (T and S) and an occupancy rate (μ) resulting in a given initial number of containers (C), the dataset includes **30 different initial configurations**.
- For all 1,440 instances, the ratio between the number of batches and C is taken to be around half, i.e., there are on average two containers per batch, which is the smallest size for a batch.

In all our experiments, the time limit is set to an hour, and the first look-ahead lower bound b_1 is used as input for both *PBFS* and *PBFSA*. All instances are solved by heuristics and lower bounds within seconds or fewer.

4.7.1 Experiment 1: Batch Model with Small Batches

Table 4.1 gives a summary of the results as follows: ✔indicates that all 30 instances are solved optimally by *PBFS*. In this case, the average solution time to solve these instances is given in seconds. Otherwise, the number of instances solved optimally is provided in red and in the form x/30. This table shows the efficiency of *PBFS* as it can solve all instances except two, for $T = 3$ and $T = 4$. Most important, the average

	T	3		4		5		6	
S	Fill rate	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent
5	C	8	10	10	13	13	17	15	20
	Solved	✔	✔	✔	✔	✔	28/30	✔	15/30
	Time (s)	0.01	0.02	0.03	0.12	0.17		5.17	
6	C	9	12	12	16	15	20	18	24
	Solved	✔	✔	✔	✔	✔	25/30	✔	14/30
	Time (s)	0.01	0.03	0.04	0.86	2.90		15.94	
7	C	11	14	14	19	18	23	21	28
	Solved	✔	✔	✔	✔	✔	24/30	23/30	5/30
	Time (s)	0.02	0.04	0.04	0.83	1.37			
8	C	12	16	16	21	20	27	24	32
	Solved	✔	✔	✔	✔	✔	20/30	22/30	5/30
	Time (s)	0.01	0.06	0.16	10.04	6.84			
9	C	14	18	18	24	23	30	27	36
	Solved	✔	✔	✔	✔	29/30	10/30	19/30	2/30
	Time (s)	0.03	0.10	0.37	8.84				
10	C	15	20	20	27	25	34	30	40
	Solved	✔	✔	✔	28/30	29/30	12/30	22/30	2/30
	Time (s)	0.03	0.10	0.54					

Table 4.1: Instances solved by *PBFS* in the batch model with small batches.

time to solve these instances is under 10 seconds for these problem sizes. Since many ports today have a maximum tier requirement of 4 and need fast solutions, *PBFS* could be used in practice in the case of small batches. However, for $T = 5$ and $T = 6$, *PBFS* cannot solve all instances optimally in a timely manner. This suggests that, as the problem grows slightly, some instances become very hard to solve, which should not be a surprise, given the $\mathcal{NP}$-hardness of the problem. To avoid such situations in real operations, heuristics can be used to provide a "good" sub-optimal solution (good in the sense of being not too far from optimality). Therefore, we want to evaluate the performance of these heuristics to know which one should be used in real operations.

We measure the performance of heuristics and the tightness of lower bounds in Tables B.1 and B.2. Concerning lower bounds, b encompasses a significant number of relocations. Adding unavoidable "bad" relocations in b_1 and b_2 improves the lower bound slightly. But experiments seem to confirm that $b_2(n) - b_1(n) \leqslant b_1(n) - b(n)$ holds, supporting our intuition that the relative increase of lower bounds $b_k(n) - b_{k-1}(n)$ decreases with k.

Concerning heuristics, EG and EM clearly outperform ERI as they result in lower expected numbers of relocations. When we have access to the optimal solutions, both heuristics are on average at most 2% more than the optimal solution. We

expect this behavior to be similar for larger instances, however we only have access to lower bounds to evaluate their performances. In this case, heuristics are on average at most 11% more than b_2, hence at most 11% from the optimal solution (even though we believe that this number is very conservative, as our lower bounds are not "tight"). Therefore, both EG and EM appear to be good solutions for the SCRP under the batch model with small batches. In this case, we recommend using EM for its simplicity of implementation and its understandability.

4.7.2 Experiment 2: Batch Model with Larger Batches

Modifying existing instances

For the sake of reproducibility, we use the existing set of instances, but slightly modify it to consider larger batches. To create these instances, for each original instance n, consider n' with the same containers in the same configuration. But, if w is the batch of a container c in n, then we take the batch of c in n' to be $w' = \left\lceil \frac{w}{\gamma} \right\rceil$, where $\gamma > 1$, i.e., we merge γ batches together. In these experiments, we take $\gamma = 2$, which implies that batches have an average size of four.

	T	3		4		5		6	
S	Fill rate	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent
5	C	8	10	10	13	13	17	15	20
	Solved	✔	✔	✔	✔	✔	21/30	✔	3/30
	Time (s)	0.08	0.29	0.14	4.55	3.20		72.70	
6	C	9	12	12	16	15	20	18	24
	Solved	✔	✔	✔	✔	✔	18/30	27/30	1/30
	Time (s)	0.08	0.47	0.25	126.37	14.74			
7	C	11	14	14	19	18	23	21	28
	Solved	✔	✔	✔	29/30	✔	9/30	14/30	0/30
	Time (s)	0.13	0.71	0.58		17.74			
8	C	12	16	16	21	20	27	24	32
	Solved	✔	✔	✔	28/30	29/30	6/30	17/30	1/30
	Time (s)	0.08	1.67	1.26					
9	C	14	18	18	24	23	30	27	36
	Solved	✔	✔	✔	26/30	25/30	5/30	15/30	0/30
	Time (s)	0.13	1.49	1.47					
10	C	15	20	20	27	25	34	30	40
	Solved	✔	✔	✔	22/30	29/30	7/30	14/30	0/30
	Time (s)	0.17	0.79	3.10					

Table 4.2: Instances solved by *PBFSA* in the batch model with larger batches.

Target error ϵ

To set our target error, we consider the following. Let n_0 be a given instance, and set $\epsilon = \frac{b(n_0)}{2}$. In this case, we know that $\epsilon \leqslant \frac{f(n_0)}{2}$, which implies that we are making an error of at most 50%. Note that this error is very conservative due to two major things: first, $b(n_0)$ is not necessarily representative of $f(n_0)$, specially if n_0 has many containers. Second, the number of samples given by Hoeffding's inequality is also very conservative, probably making our approximation substantially more accurate than what we can theoretically prove.

Results

Results are summarized in Table 4.2. Similarly to Table 4.1, ✔indicates that all 30 instances are solved approximately by *PBFSA* within the given expected error. In this case, the average solution time to solve these instances is given in seconds. Otherwise, the number of instances solved is provided in red and in the form x/30. This table shows that *PBFSA* presents several advantages. First, it solves most of instances with $T = 4$ and $S \leqslant 9$ approximately within two minutes, while we note that *PBFS* was not able to solve most of these. Moreover, as can be seen in Tables B.3 and B.4, *PBFSA* still outperforms the best heuristics despite the fact that we only set the theoretical guarantee to 50% of optimality. Together, these two advantages show the practicality of *PBFSA* for problem sizes typically encountered in real ports. Moreover, we note that increasing the batch size appears to make the problem significantly more complicated to solve as we can solve optimally larger instances in Experiment 1 (see Table 4.1). Finally, we remark that similar conclusions of Experiment 1 can be drawn for lower and upper bounds (see Tables B.3 and B.4).

4.7.3 Experiment 3: Online Model and Comparison with Ku and Arthanari [43]

Table 4.3 gives a summary similar to the two previous experiments. In addition, we report the results of [43] who set a *time limit of eight hours* for each instance. In

this table, ✓(✓) indicates that all 30 instances are solved optimally by both *PBFS* and Ku and Arthanari [43]. In this case, the average solution time in seconds to solve these instances is given for *PBFS* and for Ku and Arthanari [43] in parentheses. ✔indicates that all 30 instances are solved optimally only by *PBFS* but not in [43]. In this case, only the average solution time to solve these instances with *PBFS* is given in seconds. Otherwise, the number of instances solved by *PBFS* is provided in red and in the form x/30.

	T	3		4		5		6	
S	Fill rate	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent
5	C	8	10	10	13	13	17	15	20
	Solved	✓(✓)	✔	✓(✓)	✔	✓(✓)	28/30	✔	18/30
	Time (s)	0.01 (0.02)	0.02	0.01 (2.51)	0.09	0.16 (2483.30)		3.74	
6	C	9	12	12	16	15	20	18	24
	Solved	✓(✓)	✔	✓(✓)	✔	✔	25/30	✔	15/30
	Time	0.01 (0.01)	0.04	0.06 (139.08)	0.81	1.85		14.92	
7	C	11	14	14	19	18	23	21	28
	Solved	✓(✓)	✔	✓(✓)	✔	✔	24/30	23/30	5/30
	Time (s)	0.02 (0.33)	0.04	0.04 (207.62)	0.67	1.38			
8	C	12	16	16	21	20	27	24	32
	Solved	✓(✓)	✔	✔	✔	✔	20/30	22/30	5/30
	Time (s)	0.01 (0.33)	0.05	0.10	8.29	5.85			
9	C	14	18	18	24	23	30	27	36
	Solved	✓(✓)	✔	✔	✔	29/30	10/30	19/30	2/30
	Time (s)	0.02 (32.24)	0.09	0.38	7.26				
10	C	15	20	20	27	25	34	30	40
	Solved	✓(✓)	✔	✔	28/30	29/30	12/30	16/30	2/30
	Time (s)	0.03 (58.85)	0.08	0.52					

Table 4.3: Instances solved by *PBFS* and Ku and Arthanari [43] in the online model with small batch.

Results show strong evidence that our solution significantly improves the best existing results for the SCRP under the online model, given that we solve many larger instances optimally. Furthermore, it also outperforms the most recent algorithm in solution time for the problem sizes it can solve. It appears that, for problems for which we can solve all (or almost all) instances, most instances are "easy" to solve as the algorithm finds a solution within seconds. However, as in the batch model, there exist some instances for which the optimal solution still requires an exponential number of nodes, which makes our algorithm not tractable.

In Tables B.5 and B.6, we also report in parentheses the averages for ERI and Random found in [43]. The results for Random are consistent. However, we find

significantly better results for our implementation of ERI. This is unexpected since the only difference between the two implementations is the use of lower bound b, when the configuration has fewer than S containers remaining. Nevertheless, ERI should also be optimal in this case, as it reduces to heuristic L. So this should not affect the expected number of relocations, and we cannot explain this difference. Finally, we point out that the results are quite similar to those in Experiment 1. Indeed, the existing dataset has relatively small batches (on average 2 containers), which inherently makes the two models, batch and online, very close to each other.

4.7.4 Experiment 4: Online Model with a Unique Batch

	T	3		4		5		6	
S	Fill rate	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent	50 percent	67 percent
5	*C*	8	10	10	13	13	17	15	20
	PBFS	2.08	3.33	3.54	6.53	6.56	12.05	9.13	17.28
	L.	2.08	3.33	3.54	6.52	6.57	12.06	9.14	17.29
6	*C*	9	12	12	16	15	20	18	24
	PBFS	2.10	4.04	4.23	8.02	7.01	13.48	10.73	20.13
	L.	2.10	4.04	4.23	8.02	7.01	13.48	10.73	20.13
7	*C*	11	14	14	19	18	23	21	28
	PBFS	2.69	4.60	4.82	9.55	8.58	14.75	12.22	23.14
	L.	2.69	4.61	4.82	9.55	8.58	14.74	12.22	23.14
8	*C*	12	16	16	21	20	27	24	32
	PBFS	2.61	5.19	5.51	9.96	9.12	17.75	13.83	-
	L.	2.62	5.19	5.51	9.95	9.12	17.75	13.83	26.04
9	*C*	14	18	18	24	23	30	27	36
	PBFS	3.31	5.72	6.10	11.58	10.89	19.15	-	-
	L.	3.31	5.72	6.10	11.58	10.89	19.14	15.40	28.84
10	*C*	15	20	20	27	25	34	30	40
	PBFS	3.36	6.38	6.68	12.98	11.13	22.07	-	-
	L.	3.36	6.38	6.67	12.99	11.13	22.06	16.87	31.77

Table 4.4: Instances solved with *PBFS* and heuristic L in the online model with a unique batch.

In this experiment, we consider the existing dataset, but assign all containers into a unique batch ($W = 1$). We consider the SCRP under the online model, where containers are revealed one at a time. Note that, in this case, each container is equally likely to be retrieved, and it is equivalent to know no-information about containers relative retrieval order. For each instance, we solve it twice: first using *PBFS*, and then using heuristic L, for which we sample *10,000 scenarios* (this is different from the 5000 samples considered in previous experiments). We report the results in Table

4.4. In this table, for each problem size, we report the expected optimal number of relocations averaged over 30 instances. The notation "-" means that all 30 instances could not be solved optimally with *PBFS* within the given time limit of an hour. Note that the expected number of relocations using heuristic L reported in this experiment might be less than the one of *PBFS*; this is only due to the fact that we are sampling. Intuitively, L should be the optimal solution in this setting, and this experiment shows strong evidence that the next conjecture holds.

Conjecture 1. *Consider a configuration n with a unique batch. Let $f^o(n)$ be the minimum expected number of relocations to empty n under the online model, and let $f^{o,L}(n)$ be the expected number of relocations performed by the leveling heuristic under the online model, then*

$$f^o(n) = f^{o,L}(n). \tag{4.13}$$

This conjecture could also be made in the dynamic case, when containers arrive to be stacked. These results would have important ramifications for port operations, namely: *the optimal policy to minimize relocations, when no information is given in advance, is leveling configurations.*

Chapter 5

The Yard Crane Scheduling Problem with relocations

5.1 Contributions

Based on the literature review in Chapter 2, most studies can be partitioned into two distinct groups: the first focuses on dynamic crane scheduling for storage and retrieval requests without relocations (YCSP literature) while the second one only deals with relocations but disregards major practicalities introduced in the YCSP and the following section (for example, the third dimension of the block, the crane travel time features, the order flexibility and the dynamic nature of information).

Recently some works have started to integrates these two groups and study the storage/retrieval schedule together with relocations and storage location assignments, leading to more efficient solutions. We cited among these works [17, 80] for single crane scheduling or [57] in the case of two cranes. However, both Dell et al. [17] and Park et al. [57] consider rule-based heuristic solutions with some strong dependence on parameters that are hard to set in real operations. If Yuan et al. [80] are the first to provide an exact solution, their approach only considers a greedy optimization of crane travel time. Moreover, it appears to only apply to the special problem of steel plants which consider low stacks, hence few relocations per request. Our work contributes to the existing literature and to practice as follows:

1. We propose a model which considers together storage, retrieval and enforced relocation requests. To the best of our knowledge, this is the first work giving an exact solution for the YCS problem with relocations in the case of a single crane under realistic assumptions.

2. We introduce an objective function that jointly optimizes the current crane travel time and the expected number of future relocations.

3. We develop a model and solutions that are general and applicable to other AS/RSs where stacking occurs (such as steel plants). This model can apply to all I/O point configurations described below and potentially more; it does not assume any crane constraints and uses a detailed model for crane travel times.

4. We present more general scheduling constraints relaxing previous assumptions that all request orders were feasible. As we mention below, this is not practical in many applications where external customers are concerned as fairness becomes an issue. The flexibility model proposed in this chapter is widely applicable and could be worth studying in other settings.

5. In the setting introduced in this chapter, a solution method needs to answer two questions simultaneously: i) crane movements; and ii) storage and relocation locations assignment. To solve the YCS problem under this more realistic setting in a reasonable amount of time (e.g., a few minutes), we propose one exact algorithm and one efficient heuristic integrating both decisions.

    a. We formulate an intuitive binary IP based on crane cycles. By studying the structure of the proposed formulation, we confirm previous results which state that the complexity of the problem lies not only in the number of orders but also in the number of starting points for each request.

    b. We propose a heuristic taking advantage of theoretical properties of the previous binary IP. This solution performs a search on the feasible space of request orders and computes lower and upper bounds for each visited order. In this chapter, we use a standard local search but future work

could be done to use meta-heuristics such as simulated annealing, tabu search or genetic algorithms.

6. The last contribution of this chapter is the testing of the two solution methods both on randomly generated data and real data from a port terminal.

5.2 Problem Description

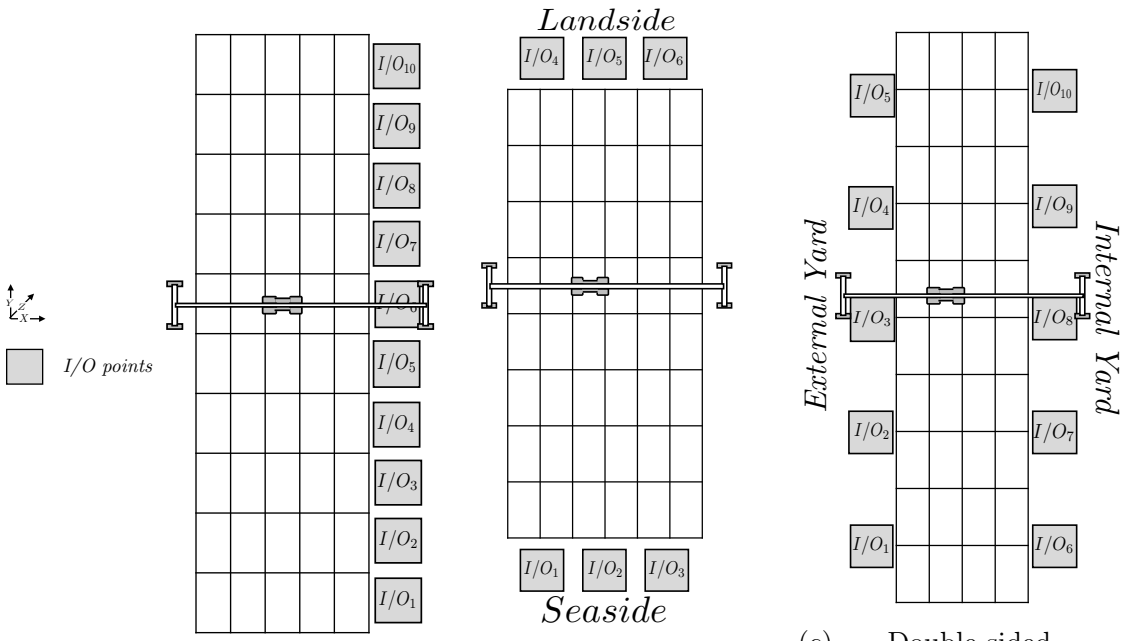
This section describes the problem of interest in this chapter. Appendix C.1 summarizes all notations defined in this section.

5.2.1 Problem Geometry

We consider the following situation. A block consists of X stacks, Y rows and Z tiers (see Figure 1-7) and we assume that this block is served by a single yard crane (YC), as shown in Figure 1-7. Each slot of the block can store a unique type of container (for example, twenty-foot or forty-foot equivalent units). Note that X is limited by the width of the crane while Z is limited by the height of the crane. Z corresponds to the maximum number of containers that can be stacked on the top of each other. Typically, these values range from 6 to 13 for X, 10 to 40 for Y and 3 to 6 for Z. Note that the tiers are counted from bottom to top. In this block, a stack s is uniquely characterized by a two-dimensional vector denoted by (s_x, s_y) corresponding to its position in the x-y dimensions. We denote by $\mathcal{S}_B$ the set of stacks in the block and note that $|\mathcal{S}_B| = X \times Y$.

We assume that there are M input/output (I/O) points around the block that we consider, denoted by I/O_m for $m \in \{1, \ldots, M\}$. These I/O points correspond to locations where vehicles, with storage or retrieval requests park. I/O points are "artificial" stacks where no container can be stored except when retrieving a container. We denote by $\mathcal{S}_I$ the set of artificial stacks corresponding to I/O points. For the sake of clarity, we denote $\mathcal{S} = \mathcal{S}_B \cup \mathcal{S}_I$ the set of all stacks.

General Automated Storage/Retrieval Systems (AS/RSs) can present many con-



(a) Asian config. (b) European config. (c) Double-sided config.

Figure 5-1: A top view of a block with three different I/O points configurations.

figurations of I/O points. In the case of port yards, Wiese et al. [75] and Carlo et al. [7] discuss the two most frequent configurations (see Figures 5-1a and 5-1b), which are commonly referred to as Asian and European style configurations. The "double-sided" configuration is another configuration of interest (see Figure 5-1c) but it has been less studied in related work. This latter configuration is mostly used in ports where internal and external yards are completely separated (for instance when the internal yard is automated or the external yard works on trains). The I/O points configuration is given as an input to the problem. In European and double-sided configurations, we denote by M_1 the number of I/O points on the seaside or internal yard side and M_2 the number of I/O points on the opposite side, such that $M = M_1 + M_2$. We mention that our solution methods are general and independent from this configuration, thus could be generally applied to other configurations.

Initially, a stack $s \in \mathcal{S}_B$ stores a certain number of containers which we denote by $z_s^i \in \{0, \ldots, Z\}$. Figure 5-3 shows these numbers in an example with Asian configuration. Note that these numbers are not given in Figure 5-1 just for the sake of clarity.

The most frequently used handling equipment in port storage yards are either rubber-tired gantry cranes (RTGs) or rail-mounted gantry cranes (RMGs). RMGs are typically automated, hence also called automated stacking cranes (ASCs). However, RTGs are more flexible as they can rotate and change blocks within the port yard (see [7] for more details on handling equipment). In this chapter, we assume that a unique YC (RTG or RMG) is allocated to the block of interest and serves requests at this block. Its initial position in the block is denoted by $s^i\,(\in \mathcal{S})$ and corresponds to the stack or the I/O point above which the crane's spreader lies (see [28, 80]). The travel time of a YC is thoroughly studied in [63] which shows that, in the case where there is no crane interference, assuming constant speed in all dimensions is capturing well the actual travel times in real operations. Figure 5-2 shows the typical movement pattern of a RMG when performing a storage or retrieval request. Each request has four phases. First, there is an empty drive from the position where the crane ended the previous request to the starting stack of the new request. According to Speer and Fischer [63], it is important to consider the time to size the spreader during empty drives. However, we disregard this time because it is required only if there are different types of containers in the block, which is not the case in this chapter. Then, the crane picks up the container with its spreader, is driven loaded to the destination stack and sets the container down. Based on this pattern, we introduce the following notations. Let $\left(v^{x,E}, v^{x,L}\right)$ be the YC trolley speed with and without load, $\left(v^{y,E}, v^{y,L}\right)$ the YC gantry speed and $\left(v^{z,E}, v^{z,L}\right)$ the YC speed to lower and hoist the spreader. We assume that all speeds are given in containers/s, i.e., how many containers can the crane pass over per second in each dimension. In addition, t^h the handling time to pick up or set down a container, which is mainly stabilization and changing direction.

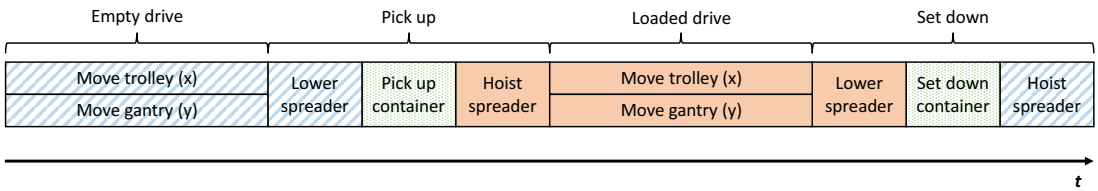


Figure 5-2: Pattern of typical YC movements for a given cycle. Striped blue indicates empty movements, solid red loaded movements and dotted green handling movements.

Using these notations, consider two stacks $s, r \in \mathcal{S}$, then the time of an empty (respectively loaded) drive of the crane from stack s to stack r can be computed as

$$t_{sr}^{(E)} = \max\left\{\frac{|s_x - r_x|}{v^{x,E}}\ ,\ \frac{|s_y - r_y|}{v^{y,E}}\right\} \text{ and } t_{sr}^{(L)} = \max\left\{\frac{|s_x - r_x|}{v^{x,L}}\ ,\ \frac{|s_y - r_y|}{v^{y,L}}\right\}.$$

The times to pick up/set down a container at tier $z \in \{1, \ldots, Z\}$ (which means that the stack has $z-1$ containers) are equal and given by

$$t^{(H)}(z) = \frac{Z+1-z}{v^{z,E}} + \frac{Z+1-z}{v^{z,L}} + t^h = \frac{2(Z+1-z)}{v^z} + t^h,$$

where $v^z = \dfrac{2}{\frac{1}{v^{z,E}} + \frac{1}{v^{z,L}}}$ is the harmonic mean of $v^{z,E}$ and $v^{z,L}$.Recall that tiers are counted from bottom to top. We assume that I/O points are equivalent to a stack with 0 containers, hence pick up/set down the container on tier 1, thus the cost of $t^{(H)}(1) = \dfrac{2Z}{v^z} + t^h$. This model is more general than the one presented in [28]. Finally, we mention that the described pattern is similar for RMGs and RTGs. The main difference between both would be the values of the parameters.

5.2.2 Requests

Given the problem geometry, the goal is to schedule storage and retrieval requests (also called productive moves) by operating the YC in a minimum amount of time. As this process is dynamic, the block sequencing approach is typically adopted to consider a more static process (see [17, 73, 28, 63, 80]). In this approach, a set of urgent requests is selected among the available ones. The selected requests are sequenced and executed by the YC. Once these are done, a new set is selected and performed. In this setting, a suitable solution should only require a few minutes to solve as the problem has to be solved repeatedly and the information about requests is not available much before it needs to be performed.

Generally, we consider a sequence of N storage and retrieval requests to perform. As accurate information is not known much in advance, N is typically small compared to $|\mathcal{S}_B|$ and usually ranges from 1 to 20. Requests are indexed based on their arrival

order. Container n and vehicle n are used to refer to the container and vehicle associated with request $n \in \{1, \ldots, N\}$. We denote by $\mathcal{N}_s$ (respectively $\mathcal{N}_r$) the indices of requests corresponding to storage requests (respectively retrieval requests) such that

$$\{1, \ldots, N\} = \mathcal{N}_s \cup \mathcal{N}_r.$$

In today's operations, requests are fulfilled solely on a first-come first-served (FCFS) basis (see [73]). On one hand, previous studies of different AS/RSs have shown that relaxing the FCFS constraint can significantly improve the overall service time. On the other hand, relaxing the FCFS policy is only possible to a certain extent to avoid issues with truck unions and maintain fairness among drivers. Consequently, *we model the flexibility of the n^{th} request ($n \in \{1, \ldots, N\}$) by two parameters $(\delta_n^-, \delta_n^+) \in \mathbb{N}^2$, such that the n^{th} request can be served between the $n - \delta_n^-$-th request and the $n + \delta_n^+$-th request.* Note that $\forall n \in \{1, \ldots, N\},\ (\delta_n^-, \delta_n^+) = (0, 0)$ means that the crane can only serve requests on a FCFS basis while $\forall n \in \{1, \ldots, N\},\ (\delta_n^-, \delta_n^+) = (n-1, N-n)$ means that all orders are feasible. This modeling assumption is further motivated by the fact that a request can either be associated with an external or internal vehicle; the latter type being owned by the port operator. Therefore, while not much flexibility can be assumed for external vehicles, the operator has full control on the flexibility of its own vehicles.

For each request $n \in \{1, \ldots, N\}$, we denote by L_n the set of stacks in which the container n can be picked up by the crane. If $n \in \mathcal{N}_s$, then L_n is the set of I/O points in which the vehicle n can park with container n. If $n \in \mathcal{N}_r$, then L_n is the stack in the block in which container n is stored. We denote E_n to be the set of stacks onto which container n can be put down. Typically, to have as much flexibility as possible, we will consider $E_n = \mathcal{S}_B$ for $n \in \mathcal{N}_s$ and $E_n = \mathcal{S}_I$ for $n \in \mathcal{N}_r$, thus generalizing settings in [73, 28].

As we mentioned, for each retrieval request $n \in \mathcal{N}_r$, $L_n = \{s_n\}$ corresponds to the stack in the block in which container n is stored. In addition, we must be given the exact tier of stack s_n in which container n is stored. We denote this tier by

$z_n \in \{1, \ldots, z^i_{s_n}\}$. If container n is on the top of its stack, i.e., $z_n = z^i_{s_n}$, then it can be retrieved directly. However, for a significant number of requests, container n is blocked by other containers, i.e., $z_n < z^i_{s_n}$, then the YC has to relocate containers blocking containers n from stack s_n to another stack of the block. These container moves are called unproductive requests (also called relocations or reshuffles). They result both from a lack of information for future requests and inefficient decisions in past operations of the YC. Minimizing these unproductive requests has been an important metric for port operators (see [27]).

We denote by $\mathcal{N}_u$ the set of unproductive requests needed to perform all retrieval requests in $\mathcal{N}_r$. Therefore, the YC effectively has $\overline{N} \geqslant N$ requests to perform where

$$\{1, \ldots, \overline{N}\} = \mathcal{N}_s \cup \mathcal{N}_r \cup \mathcal{N}_u.$$

Naturally, we can extend the notations L_n, E_n, s_n and z_n to each relocation request $n \in \mathcal{N}_u$. If $L_n = \{s_n\}$, then s_n is the stack where the blocking container n is stored and z_n is the tier of container n. Typically, we will consider $E_n = \mathcal{S}_B \setminus L_n$, which means that container n could be relocated anywhere except where it is already. Given these $\overline{N}$ requests, we define

$$\mathcal{S}^{(L)} = \bigcup_{n \in \{1,\ldots,\overline{N}\}} L_n \ , \ \mathcal{S}^{(E)} = \bigcup_{n \in \{1,\ldots,\overline{N}\}} E_n,$$

such that $\mathcal{S}^{(L)}$ is the set of starting stacks for loaded drives of the YC, while $\mathcal{S}^{(E)}$ is the set of starting stacks for empty drives of the YC. Finally, we define $\mathcal{S}_R = \mathcal{S}_B \cap \mathcal{S}^{(L)}$ as the set of stacks of the block where there is at least one container that needs to be retrieved.

For each request $n \in \mathcal{N}_r \cup \mathcal{N}_u$, b_n denotes the index of the container directly blocking container n (where $b_n = 0$ means that container n is on the top of its stack).

Finally, consider a stack $s \in \mathcal{S}_B$, we let m_s denote the lowest container to be

retrieved in s. Using this notation, we define $\widetilde{z}_s$ such that

$$\widetilde{z}_s = \begin{cases} z_{m_s} - 1, & \text{if } s \in \mathcal{S}_R, \\ z_s^i, & \text{otherwise.} \end{cases}.$$

Here, $\widetilde{z}_s$ represents the number of containers in s after all containers have been retrieved and none has been stored or relocated. Note that it is also the minimum number of containers stack s can have after performing all $\overline{N}$ requests.

As in [80], we assume that *the number of cycles (empty/loaded drives) done by the YC to perform all $\overline{N}$ requests is exactly $\overline{N}$*. Consider a stack $s \in \mathcal{S}_R$ where there is at least one container that needs to be retrieved. This assumption prevents any containers to be relocated or stored on stack s before m_s (the lowest container to be retrieved in s) has been retrieved. Even though this assumption seems to be restrictive, each cycle of the crane takes a significant amount of time. So, relocating a container twice for the same set of requests is inefficient. Moreover, we note that this assumption is realistic due to the fact that N, hence $\overline{N}$ are small compared to $|\mathcal{S}_B|$, i.e., there always exists stacks $\notin \mathcal{S}_R$ where stacking and relocation requests can be done. However, this also prevents a container to be moved twice before it is retrieved and potentially makes the problem infeasible. To insure that the FCFS policy is always feasible under this assumption and regardless of the level of flexibility of both requests, we assume the following: consider the case where two retrieval requests $n, n' \in \mathcal{N}_r$ are such that containers n and n' are stored in the same stack (i.e., $L_n = L_{n'}$) and no containers in between needs to be retrieved. If container n lies above container n', i.e., $z_n > z_{n'}$, then we assume that $n < n'$. Indeed, it is common practice for two trucks requesting containers in the same stack to change their orders: if the truck waiting for the upper container arrived after the truck waiting for the lower container, then the truck waiting for the upper container "skips" the line and gets served just before the truck waiting for the lower container.

Example. Figure 5-3 shows a top view of a small block with Asian configuration ($X = 5$, $Y = 10$, $Z = 4$ and $M = 10$) and each stack shows the initial number of con-

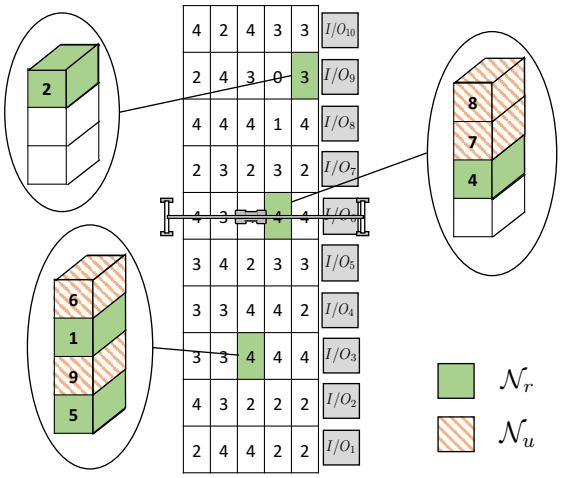


Figure 5-3: A top view of a block with Asian configuration. The integer in each stack of the block corresponds to the number of containers currently stored in the stack. For stacks in $\mathcal{S}_R$, we highlight containers to be retrieved ($\mathcal{N}_r$) and relocated ($\mathcal{N}_u$).

tainers $(z_i^s)_{s\in\mathcal{S}_B}$. The crane starts above stack $s^i = (3, 5)$. There are $N = 5$ productive requests, a single storage and four retrievals. We consider that only FCFS is possible, i.e., $(\delta_n^-, \delta_n^+) = (0, 0)$. We have $\mathcal{N}_s = \{3\}$ and the storage request can be done from any I/O point, i.e., $L_3 = \mathcal{S}_I$ and $E_3 = \mathcal{S}_B$. The retrieval requests ($\mathcal{N}_r = \{1, 2, 4, 5\}$) are associated with containers shown in solid green in Figure 5-3. Containers to be retrieved are located in stacks $(L_1, L_2, L_4, L_5) = (\{(3, 8)\}, \{(5, 2)\}, \{(4, 5)\}, \{(3, 8)\})$ and tiers $(z_1, z_2, z_4, z_5) = (3, 3, 2, 1)$. These requests require 4 relocations (shown with red stripped containers) such that $\mathcal{N}_u = \{6, 7, 8, 9\}$ and $\overline{N} = 9$. In this example, we have $\mathcal{S}_R = \{(5, 2), (4, 5), (3, 8)\}$, $\left(\min_{(5,2)}, \min_{(4,5)}, \min_{(3,8)}\right) = (2, 4, 5)$ and $(b_1, b_2, b_4, b_5, b_6, b_7, b_8, b_9) = (6, 0, 7, 9, 0, 8, 0, 1)$. The goal is to provide the route of the YC to perform all these requests and assign new stacks to stored and relocated containers.

5.2.3 Objective Function

When scheduling all $\overline{N}$ requests, the main goal is typically to minimize the crane travel time and make decisions about storage and relocation locations accordingly. However, as we previously mentioned, the number of relocations for future requests, hence the crane travel time for future requests is directly impacted by current storage

and relocation decisions. Because our problem is dynamic, we want to optimize both our current and future crane travel times. However, these are naturally conflicting objectives. We explain and formalize this trade-off in the following objective function.

On one hand, the *immediate objective* refers to the crane travel time to perform the $\overline{N}$ requests. Minimizing the immediate objective means that containers involved in stacking and relocation requests should be stored in the closest stacks to the crane where a slot is available, potentially creating higher stacks.

On the other hand, the *cost-to-go* relates to the future crane travel time. Since we do not assume any information about future requests, the variability in expected crane travel time is mostly correlated with the number of future relocations. It has been shown when only considering retrievals that the expected number of blocking containers is a good proxy for the number of relocations when the number of stacks is large (see [25]). Moreover, this metric also makes sense from a practical point of view. The widely used leveling heuristic minimizes the number of blocking containers and appears to be optimal experimentally with respect to the expected number of relocations when requests come one at a time (see [24]). Finally, this metric has the advantage to require only the number of containers per stack and avoids creating high stacks. From now on, the *cost-to-go function is taken to be the expected number of blocking containers in the block.*

Formally, we define α_z to be the expected number of blocking containers in a stack of z containers in the case where no information is available. From [25] (and Section 3.4), we have $\alpha_0 = 0$ and

$$\alpha_z = z - \sum_{i=1}^{z} \frac{1}{i}, \ \forall z \in \{1, \dots, Z\}. \tag{5.1}$$

Now, consider that after performing all $\overline{N}$ requests, the height of stack $s \in \mathcal{S}_B$ is denoted by $z_s^f \in \{0, \dots, Z\}$. Then, using the previous notation, the cost to go which is the expected total number of blocking containers can be computed as

$$\textit{cost-to-go} = \sum_{s \in \mathcal{S}_B} \alpha_{z_s^f}. \tag{5.2}$$

Let $\gamma \geqslant 0$ be the importance/conversion factor between future relocations and current crane travel time. Using a classic scalarization technique, by minimizing

$$\textit{Objective function} = \textit{immediate cost} + \gamma \times \textit{cost-to-go}, \tag{5.3}$$

we balance the objective between greedily minimizing immediate cost and minimizing the cost-to-go. Note that $\gamma = 0$ means that we only minimize immediate cost, while $\gamma \to \infty$ implies minimizing the expected number of relocations, hence it is equivalent to the leveling heuristic.

5.3 Binary Integer Program and Theoretical Properties

This section presents an exact method to solve the YSC problem with storage and relocation location assignments. To formulate the problem as a binary IP, we first describe the variables, then translate each constraint into linear equalities/inequalities using these variables, and finally compute the objective function as a linear function of the variables. Based on this formulation, we show that the integrality constraints of a significant portion of the variables can be relaxed under Condition (A) for γ, described in page 16. Before stating the mathematical formulation, we present concisely the main notations of Section 5.2:

- $\diamond$ (X, Y, Z): dimensions of the block
- $\diamond$ N: number of productive requests, indexed by arrival order.
- $\diamond$ $\mathcal{N}_s$: indices corresponding to storage requests.
- $\diamond$ $\mathcal{N}_r$: indices corresponding to retrieval requests.
- $\diamond$ $\overline{N}$: total number of requests to perform all N productive requests.
- $\diamond$ $\mathcal{N}_u$: indices corresponding to unproductive requests.

For each request $n \in \{1, \ldots, \overline{N}\}$, we are given:

- L_n: set of stacks from which container n can be picked up by the crane.
- E_n: set of stacks onto which container n can be put down by the crane.
- (δ_n^-, δ_n^+): flexibility of request n.
- b_n: container directly blocking container n. If n is on top of its stack, $b_n = 0$.

There are several stacks of interest:

- s^i: initial position of the crane.
- $\mathcal{S}_B$: set of stacks in the block.
- $\mathcal{S}_R$: set of stacks in the block where there is at least one container to be retrieved.
- $\mathcal{S}^{(L)}$: set of stacks from which the crane can start a loaded drive.
- $\mathcal{S}^{(E)}$: set of stacks from which the crane can start an empty drive.

Finally, we are given:

- m_r: lowest container to be retrieved in stack $r \in \mathcal{S}_R$.
- $\widetilde{z}_r$: number of containers in stack $r \in \mathcal{S}_B$ after all containers have been retrieved and none has been stored or relocated.
- $t_{sr}^{(L)}$: cost of a loaded drive of the crane from stack $s \in \mathcal{S}^{(L)}$ to stack $r \in \mathcal{S}^{(E)}$.
- $t_{rs}^{(E)}$: cost of an empty drive of the crane from stack $r \in \mathcal{S}^{(E)}$ to stack $s \in \mathcal{S}^{(L)}$.
- γ: weight on the cost-to-go.
- α_z: expected number of blocking containers in a stack with z containers.
- v^z: harmonic mean of the vertical speeds with and without containers.

5.3.1 Formulation

Variables

The mathematical formulation introduced in this chapter uses the following binary variables ($\in \{0,1\}$).

- w_{nsk} indicates that container n is moved (retrieved, stored or relocated) from stack s during crane cycle k. This type of variable is defined $\forall\ n \in \left\{1,\ldots,\overline{N}\right\}$, $\forall\ s \in L_n$, $\forall\ k \in \left\{1,\ldots,\overline{N}\right\}$.

- d_s^i indicates that the crane has an empty drive from its initial position s^i to stack s during the first cycle. These are defined $\forall\ s \in \mathcal{S}^{(L)}$.

- $d_{rsk}^{(E)}$ indicates that the crane has an empty drive from stack r to stack s during its k^{th} cycle. These variables are defined $\forall\ r \in \mathcal{S}^{(E)},\ \forall\ s \in \mathcal{S}^{(L)},\ \forall\ k \in \left\{2,\ldots,\overline{N}\right\}$.

- $d_{srk}^{(L)}$ indicates that the crane has a loaded drive from stack s to stack r during its k^{th} cycle , and is defined $\forall\ s \in \mathcal{S}^{(L)},\ \forall\ r \in \mathcal{S}^{(E)},\ \forall\ k \in \left\{1,\ldots,\overline{N}\right\}$.

- f_{rz} indicates that there are z containers in stack r after the $\overline{N}$ requests are performed. This last type of variable is defined $\forall\ r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B,\ \forall\ z \in \left\{\widetilde{z}_r,\ldots,Z\right\}$. Indeed, a stack can receive a stored or relocated container only if it is both a potential ending stack for loaded drives (i.e., $\mathcal{S}^{(E)}$) and in the block (i.e., $\mathcal{S}_B$). In addition, $\widetilde{z}_r$ is defined to be a lower bound on the final number of containers in stack r.

Constraints

First, we denote by D the set of variables $\left(d^i, d^{(E)}, d^{(L)}, f\right)$. We denote the feasible polyhedron of our problem by $\mathcal{P}$ such that our feasible set is by definition $(w, D) \in \mathcal{P} \cap \{0,1\}$. We decompose $\mathcal{P}$ as follows:

$$\mathcal{P} = \mathcal{W} \cap \mathcal{D} \cap \mathcal{L},$$

where $\mathcal{W}$ is a polyhedron corresponding to constraints involving only w variables, and $\mathcal{D}$ to constraints involving only variables in D. Finally, $\mathcal{L}$ is the polyhedron corresponding to constraints linking w and the variables in D (in particular $d^{(L)}$). We describe these three polyhedra in detail.

The polyhedron $\mathcal{W}$. This polyhedron is defined by three types of constraints. We say $w \in \mathcal{W}$ if it verifies Equations (5.4)-(5.6).

Assignment of requests to crane cycles – Each container must be moved (delivered, stored or relocated) during a unique crane cycle and each crane cycle must perform exactly one request:

$$\forall\ n \in \{1, \ldots, \overline{N}\}, \quad \sum_{\substack{s \in L_n \\ k \in \{1,\ldots,\overline{N}\}}} w_{nsk} \ = 1, \tag{5.4a}$$

$$\forall\ k \in \{1, \ldots, \overline{N}\}, \quad \sum_{\substack{n \in \{1,\ldots,\overline{N}\} \\ s \in L_n}} w_{nsk} \ = 1. \tag{5.4b}$$

Precedence constraints – Container $n \in \mathcal{N}_r \cup \mathcal{N}_u$ associated with a retrieval or relocation request, cannot be moved during cycle $k \in \{1, \ldots, \overline{N}\}$, if there is a container blocking it ($b_n \neq 0$), and container b_n has not been previously moved (e.g., containers 4 and 9 in Figure 5-3):

$$\forall\ n \in \{n' \in \mathcal{N}_r \cup \mathcal{N}_u \mid b_{n'} \neq 0\},\ \forall\ k \in \{1, \ldots, \overline{N}\}, \sum_{s \in L_n} w_{nsk} - \sum_{\substack{s \in L_{b_n} \\ k' \in \{1,\ldots,k-1\}}} w_{b_n s k'} \ \leqslant 0, \tag{5.5}$$

where the second sum is 0 when $k = 1$.

Order constraints – Recall that by definition of the flexibility of a productive request $n \in \{1, \ldots, N\}$, this request has to be served between the $n - \delta_n^-$-th productive request and the $n + \delta_n^+$-th productive request. An equivalent reformulation is that productive request n can have at most $n + \delta_n^+ - 1$ productive requests served before it, and at most $N - n + \delta_n^-$ productive requests after it, which is how the last constraints

are formulated below:

$$\forall\, n \in \{1,\dots,N\},\ \forall\, k \in \{n+\delta_n^+ + 1,\dots,\overline{N}\},$$
$$\sum_{\substack{n'\in\{1,\dots,N\}\\ s'\in L_{n'}\\ k'\in\{1,\dots,k-1\}}} w_{n's'k'} + \left(k - \left(n+\delta_n^+\right)\right) \times \sum_{s\in L_n} w_{nsk} \ \leqslant k-1, \tag{5.6a}$$

$$\forall\, n \in \{1,\dots,N\},\ \forall\, k \in \{1,\dots,\overline{N} - (N-n+\delta_n^-) - 1\},$$
$$\sum_{\substack{n'\in\{1,\dots,N\}\\ s'\in L_{n'}\\ k'\in\{k+1,\dots,\overline{N}\}}} w_{n's'k'} + \left(\overline{N} - k - \left(N-n+\delta_n^-\right)\right) \times \sum_{s\in L_n} w_{nsk} \ \leqslant \overline{N} - k. \tag{5.6b}$$

The polyhedron $\mathcal{D}$. This polyhedron is defined by six types of constraints. We say $D \in \mathcal{D}$ if it verifies Equations (5.7)-(5.12).

Uniqueness of empty drive – The crane can only have one empty drive for each cycle (the first being slightly different as the cycle has to start at the initial position of the crane s^i):

$$\sum_{s\in S^{(L)}} d_s^i \ = 1, \tag{5.7a}$$

$$\forall\, k \in \{2,\dots,\overline{N}\},\ \sum_{\substack{r\in\mathcal{S}^{(E)}\\ s\in\mathcal{S}^{(L)}}} d_{rsk}^{(E)} \ = 1. \tag{5.7b}$$

Uniqueness of loaded drive – Similarly, the crane is only allowed one loaded drive for each cycle:

$$\forall\, k \in \{1,\dots,\overline{N}\},\ \sum_{\substack{s\in\mathcal{S}^{(L)}\\ r\in\mathcal{S}^{(E)}}} d_{srk}^{(L)} \ = 1. \tag{5.8}$$

"Conservation of flow" after an empty drive – During crane cycle k, if the crane empty drive ends in stack s to pick up a container, then the loaded drive for this cycle

should start from stack s:

$$\forall s \in \mathcal{S}^{(L)}, \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{sr1} - d^{i}_{s} = 0, \tag{5.9a}$$

$$\forall s \in \mathcal{S}^{(L)}, \ \forall\, k \in \{2, \ldots, \overline{N}\}, \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{srk} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{rsk} = 0. \tag{5.9b}$$

"Conservation of flow" after a loaded drive – Similarly, if during crane cycle $k-1$, the crane loaded drive ends in stack r, then the crane empty drive for cycle k should start from stack r:

$$\forall r \in \mathcal{S}^{(E)}, \ \forall\, k \in \{2, \ldots, \overline{N}\}, \sum_{s \in \mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = 0. \tag{5.10}$$

Uniqueness of final number of containers – Because the final number of containers in stack r is encoded into binary variables, we need to insure that only one integer (i.e., one variable) is selected:

$$\forall\, r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1. \tag{5.11}$$

Final number of containers – For each stack r in the block where a loaded drive of the crane can end, the final number of containers in stack r can be computed directly as $\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz}$. This term has to be equal to the sum of two other terms: the number of containers in r after all containers have been retrieved and before any container is stored or relocated (i.e., $\widetilde{z}_r$) and the total number of crane loaded drives ending on stack r over the $\overline{N}$ crane cycles. Thus,

$$\forall\, r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1, \ldots, \overline{N}\}}} d^{(L)}_{srk} = \widetilde{z}_r. \tag{5.12}$$

The polyhedron $\mathcal{L}$. Finally, we describe the two types of inequalities defining $\mathcal{L}$ that involve variables w and D ($d^{(L)}$ specifically). We say that $(w, D) \in \mathcal{L}$ if (w, D) verify Equations (5.13)-(5.14).

Enforcing loaded drive – If container n is moved from stack s during crane cycle k, then the crane loaded drive of cycle k must start from s and end in a stack in E_n:

$$\forall\ n \in \{1,\ldots,\overline{N}\},\ \forall\ s \in L_n,\ \forall\ k \in \{1,\ldots,\overline{N}\},\ \sum_{r\in E_n} d^{(L)}_{srk} - w_{nsk}\ \geqslant 0. \tag{5.13}$$

Precedence relation between last retrieval and storage/relocation location assignments – Based on our assumption in Section 5.2, if r is a stack from which there is at least one retrieval request, then we assume that until m_r has been retrieved, no container can be stored or relocated to stack r (recall that m_r is the lowest container in stack r that needs to be retrieved):

$$\forall\ r \in \mathcal{S}_R,\ \forall\ k \in \{1,\ldots,\overline{N}\},\ \sum_{s\in\mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k'\in\{1,\ldots,k-1\}} w_{m_r rk'}\ \leqslant 0. \tag{5.14}$$

Objective function

We now formulate Equation (5.3) as a linear function of the previous binary variables. We use the fact if z^f_r denotes the number of containers in stack r after all $\overline{N}$ requests, then for any function $h(.)$ we have $h\left(z^f_r\right) = \sum_{z\in\{\widetilde{z}_r,\ldots,Z\}} h(z) f_{rz}$. Moreover, if r is a stack in the block but where no crane loaded drive can end, then its final number of containers is necessarily the number of containers after all retrievals have been done, i.e., if $r \in \mathcal{S}_B \setminus \mathcal{S}^{(E)}$, then $z^f_r = \widetilde{z}_r$. Using these two observations, the cost-to-go has a constant part equal to $\sum_{r\in\mathcal{S}_B\setminus\mathcal{S}^{(E)}} \alpha_{\widetilde{z}_r}$ and a variable part which can be expressed as:

$$\sum_{\substack{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B \\ z\in\{\widetilde{z}_r,\ldots,Z\}}} \alpha_z f_{rz}, \tag{5.15}$$

where α_z is defined in Equation (5.1).

Now we focus on expressing the immediate cost, which is the total travel time of the crane to perform the $\overline{N}$ requests. As explained in Figure 5-2, each cycle can be decomposed in four phases (empty drive, pick-up, loaded drive and put-down), which we now express mathematically:

Empty drives – For the first cycle, it is identified by variables d^i_r. By definition the empty first drive starts at stack s^i such that the cost of the first empty drive is given by

$$\sum_{r \in \mathcal{S}^{(L)}} t^{(E)}_{s^i r} d^i_r. \tag{5.16}$$

For other cycles, the empty drives are indicated by variables $d^{(E)}_{rsk}$, each with costs $t^{(E)}_{rs}$, so the cost of all other empty drives is equal to

$$\sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)} \\ k \in \{2,\dots,\overline{N}\}}} t^{(E)}_{rs} d^{(E)}_{rsk}. \tag{5.17}$$

Loaded drives – Similarly to the empty drives, the cost of all loaded drives can be expressed as

$$\sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)} \\ k \in \{1,\dots,\overline{N}\}}} t^{(L)}_{sr} d^{(L)}_{srk}. \tag{5.18}$$

Pick-ups and put-downs – As we pointed out in Section 5.2, both these operations are assumed to have the same cost structure (one loaded vertical move, one empty vertical move and one handling time). Thus we compute their total cost jointly as follows.

Depending on the type of the request that is performed, these costs can either be constant (i.e., independent on the variables) or not. For $n \in \mathcal{N}_r$, both pick up and put down costs are constant. Indeed, the pick-up has to be done from tier z_n and the container has to be put down at one I/O point (on the floor, i.e., tier 1). Thus, the cost of the pick-up is $t^{(H)}(z_n)$ and the cost of put-down is $t^{(H)}(1)$. For $n \in \mathcal{N}_s$, the pick-up has to occur at one I/O point so the cost of pick-up is constant, i.e., $t^{(H)}(1)$. However, the cost of putting down a stored container depends on the selected stack, hence is variable. Let us denote $v(n)$ the tier at which container n is stored, then the cost of putting down a stored container is given by $t^{(H)}(v(n))$. Similarly, for $n \in \mathcal{N}_u$, the cost of pick-up is constant, i.e., $t^{(H)}(z_n)$ but the cost of putting down a relocated container depends on the selected stack, i.e., $t^{(H)}(v(n))$ if $v(n)$ the tier at

which container n is relocated. We summarize this analysis in Table 5.1.

n	$\mathcal{N}_r$	$\mathcal{N}_s$	$\mathcal{N}_u$
Pick-up cost	$t^H(z_n)$	$t^H(1)$	$t^H(z_n)$
Put-down cost	$t^H(1)$	$\boldsymbol{t^H(v(n))}$	$\boldsymbol{t^H(v(n))}$

Table 5.1: Pick-up and put-down costs for different types of requests. Terms in bold identify the variable costs.

In summary, the variable part is

$$\sum_{n\in\mathcal{N}_s\cup\mathcal{N}_u} t^{(H)}\left(v(n)\right).$$

Recall that we have $t^{(H)}\left(v(n)\right) = \dfrac{2(Z+1-v(n))}{v^z} + t^h$. Therefore, there is a constant part computed as $C = \sum_{n\in\mathcal{N}_r\cup\mathcal{N}_u} t^{(H)}\left(z_n\right) + |\mathcal{N}_r\cup\mathcal{N}_s| \times \times t^{(H)}\left(1\right) + |\mathcal{N}_s\cup\mathcal{N}_u| \times t^{(H)}\left(0\right)$, while the variable part can be expressed as

$$-\frac{2}{v^z} \times \sum_{n\in\mathcal{N}_s\cup\mathcal{N}_u} v(n).$$

The next lemma shows how to express this cost in terms of the final number of containers per stack, i.e., $\left(z_r^f\right)_{r\in\mathcal{S}_B}$ (the proof is in Appendix).

Lemma 11. *Let $n \in \mathcal{N}_s \cup \mathcal{N}_u$. Let $v(n)$ be the tier at which container n is stored or relocated when performing request n and z_r^f the number of containers in stack r after performing all $\overline{N}$ requests, then we have*

$$\sum_{n\in\mathcal{N}_s\cup\mathcal{N}_u} v(n) = \frac{1}{2}\sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} z_r^f\left(z_r^f+1\right) - \frac{1}{2}\sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \widetilde{z}_r\left(\widetilde{z}_r+1\right).$$

Using Lemma 11, we add the second term to the constant to get $C' = C + \dfrac{1}{v^z}\sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \widetilde{z}_r\left(\widetilde{z}_r+1\right)$. Using the same observation as that for the cost-to-go, the

variable part of the cost to pick up and put down all containers is:

$$-\sum_{\substack{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B\\ z\in\{\widetilde{z_r},\dots,Z\}}}\frac{1}{v^z}z\left(z+1\right)f_{rz}. \tag{5.19}$$

Combining Equations (5.16)-(5.19), we can write the objective function as

$$\sum_{s\in\mathcal{S}^{(L)}}t^{(E)}_{s^i s}d^i_s+\sum_{\substack{r\in\mathcal{S}^{(E)}\\ s\in\mathcal{S}^{(L)}\\ k\in\{2,\dots,\overline{N}\}}}t^{(E)}_{rs}d^{(E)}_{rsk}+\sum_{\substack{s\in\mathcal{S}^{(L)}\\ r\in\mathcal{S}^{(E)}\\ k\in\{1,\dots,\overline{N}\}}}t^{(L)}_{sr}d^{(L)}_{srk}+\sum_{\substack{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B\\ z\in\{\widetilde{z_r},\dots,Z\}}}\beta_z f_{rz}, \tag{5.20}$$

where

$$\beta_z=\gamma\times\alpha_z-\frac{1}{v^z}z\left(z+1\right)=\gamma\times\left(z-\sum_{i=1}^{z}\frac{1}{i}\right)-\frac{1}{v^z}z\left(z+1\right). \tag{5.21}$$

Note that the objective function defined in Equation (5.20) only depends on variables in D and not w. Thus for the sake of clarity, we can also express this objective function as $c^T D$ where c are the corresponding costs for each variable in D.

Condition (A). For the remaining of the chapter, we say that γ verifies Condition (A) if

$$\gamma>\frac{2Z(Z-1)}{v^z}. \tag{5.22}$$

Lemma 12. *Let γ verify Condition (A), $z, z_1, z_2\in\{0,\dots,Z\}$ such that $z_2<z<z_1$, then we have*

$$\frac{z-z_2}{z_1-z_2}\beta_{z_1}+\frac{z_1-z}{z_1-z_2}\beta_{z_2}>\beta_z.$$

Summary of formulation

In conclusion, we can formulate our problem as

$$\min_{(w,D)\in\mathcal{P}\cap\{0,1\}}\left(c^T D\right).$$

Using the decomposition of $\mathcal{P}$, this formulation can also be written as

$$\min_{(w,D)\in\{0,1\}} \left(c^T D\right) \qquad \text{Equation (5.20)}$$

$$s.t. \begin{cases} w \in \mathcal{W} & \text{Equations (5.4)-(5.6)} \\ D \in \mathcal{D} & \text{Equations (5.7)-(5.12)} \\ (w, D) \in \mathcal{L} & \text{Equations (5.13)-(5.14)} \end{cases}$$

5.3.2 Relaxation of Integrality Conditions

Definition 1. *Let P and P' be two optimization problems. We say that P and P' are equivalent if there exists a transformation from any optimal solution of P to an optimal solution of P' and vice versa.*

Using the structure of the previous mathematical formulation, we now prove that, if γ verifies Condition (A), we can relax the integrality constraints for variables in D and still get an integral solution. The process has two steps: first, given some $w \in \mathcal{W} \cap \{0,1\}$, we formulate the subproblem as an equivalent binary IP that has a simpler structure. Then, we show that, given that γ verifies Condition (A), any optimal extreme point of the relaxation of the simpler formulation is integral. This implies that, given $w \in \mathcal{W} \cap \{0,1\}$, the subproblem can be solved in polynomial time as we just have to solve a linear program. Formally, let us consider

$$\forall\ w \in \mathcal{W},\ \mathcal{L}(w) = \{D \in \mathcal{D} \mid (w, D) \in \mathcal{L} \text{ and } 0 \leqslant D \leqslant 1\}.$$

Using this definition, we can re-write the original problem as:

$$\min_{w\in\mathcal{W}\cap\{0,1\}} \left(\min_{D\in\mathcal{L}(w)\cap\{0,1\}} \left(c^T D\right) \right).$$

Let $w \in \mathcal{W} \cap \{0,1\}$, consider $\Pi(w)$ to be the subproblem associated with w and defined as

$$\min_{D\in\mathcal{L}(w)\cap\{0,1\}} \left(c^T D\right).$$

In the first step, we reformulate $\Pi(w)$ in an equivalent subproblem denoted by $\overline{\Pi}(w)$

and defined as

$$\min_{\overline{D}\in\overline{\mathcal{L}}(w)\cap\{0,1\}} \left(\overline{c}^T\overline{D}\right),$$

such that $\overline{\Pi}(w)$ is simpler to analyze than $\Pi(w)$. Under Condition (A), we show that any optimal extreme point of $\overline{\mathcal{L}}(w)$ is integral. Consequently, $\overline{\Pi}(w)$ is equivalent to its linear programming relaxation denoted by $\Pi^L(w)$ and defined as

$$\min_{\overline{D}\in\overline{\mathcal{L}}(w)} \left(\overline{c}^T\overline{D}\right).$$

In conclusion, since the linear program $\Pi^L(w)$ is equivalent to $\Pi(w)$, then $\Pi(w)$ can be solved in polynomial time.

Subproblem reformulation

Let $w \in \mathcal{W} \cap \{0,1\}$, we now define problem $\overline{\Pi}(w)$ equivalent to $\Pi(w)$ but with a simpler structure. To do so, we define some notations:

$$\forall\ k \in \{1,\dots,\overline{N}\},\ (\nu_k,\sigma_k) = \left\{(n,s) \in \{1,\dots,\overline{N}\} \times \mathcal{S}^{(L)}\ \middle|\ s \in L_n \text{ and } w_{nsk} = 1\right\}. \quad (5.23)$$

Here (ν_k,σ_k) represent the indices of the request performed at stage k and the stack from which this request is performed. Note that these are clearly unique for each $k \in \{1,\dots,\overline{N}\}$ since $w \in \mathcal{W} \cap \{0,1\}$. Using these notations, we can define

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \overline{E}_{\nu_k} = \left\{ r \in E_{\nu_k} \ \middle|\ \begin{array}{c} (r \notin \mathcal{S}_R)\ \ or \\ (r \in \mathcal{S}_R \text{ and } \exists\ k' \in \{1,\dots,k-1\} \text{ s.t. } \nu_{k'} = m_r) \end{array} \right\} \quad (5.24)$$

where $\overline{E}_{\nu_k}$ is a subset of stacks in E_{ν_k} where a request can end. In addition, it disregards stacks $r \in \mathcal{S}_R$ for which m_r has not been retrieved before stage k. This leads to

$$\overline{\mathcal{S}}_B = \mathcal{S}^{(E)} \cap \mathcal{S}_B \cap \bigcup_{k\in\{1,\dots,\overline{N}\}} \overline{E}_{\nu_k}, \quad (5.25)$$

with $\mathcal{S}_B$ the set of stacks where requests can end given w. Only these stacks are going to have a number of containers that is different from $\widetilde{z}_r$. Finally, we consider

$$\forall\ r \in \overline{\mathcal{S}}_B,\ \overline{K}_r = \left\{ k \in \{1, \ldots, \overline{N}\} \ \middle|\ r \in \overline{E}_{\nu_k} \right\}, \tag{5.26}$$

which corresponds to the set of stages where a container can be stored or relocated to stack r. Based on these notations, we consider $\overline{\Pi}(w)$ to be the following optimization problem

$$\left\{ \begin{array}{l} \min\limits_{(\overline{d},\overline{f})\in\{0,1\}} \left(\sum\limits_{\substack{k\in\{1,\ldots,\overline{N}\} \\ r\in\overline{E}_{\nu_k}}} \overline{t}_{rk}\overline{d}_{rk} + \sum\limits_{\substack{r\in\overline{\mathcal{S}}_B \\ z\in\{\widetilde{z}_r,\ldots,Z\}}} \beta_z \overline{f}_{rz} \right) \\ \\ s.t. \left\{ \begin{array}{ll} \forall\ k \in \{1, \ldots, \overline{N}\}, & \sum\limits_{r\in\overline{E}_{\nu_k}} \overline{d}_{rk} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B, & \sum\limits_{z\in\{\widetilde{z}_r,\ldots,Z\}} \overline{f}_{rz} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B, & \sum\limits_{z\in\{\widetilde{z}_r,\ldots,Z\}} z\overline{f}_{rz} - \sum\limits_{k\in\overline{K}_r} \overline{d}_{rk} = \widetilde{z}_r, \end{array} \right. \end{array} \right.$$

where

$$\forall\ k \in \{1, \ldots, \overline{N}\}, \forall\ r \in \overline{E}_{\nu_k},\ \overline{t}_{rk} = \left\{ \begin{array}{ll} t^{(L)}_{\sigma_k r} + t^{(E)}_{r\sigma_{k+1}} & \text{if } k < \overline{N}, \\ t^{(L)}_{\sigma_{\overline{N}} r} & \text{otherwise.} \end{array} \right.$$

Note that this problem depends on w through ν which define $\overline{E}_{\nu_k}$ and $\overline{S}_B$, as well as σ defining $\overline{t}$.

For the sake of clarity, let us define $\overline{D} = \left(\overline{d}, \overline{f}\right)$ and $\overline{c}$ as the associated cost in $\overline{\Pi}(w)$. Finally, $\overline{\mathcal{L}}(w)$ be the feasible set of $\overline{\Pi}(w)$ without the integrality constraints. The next lemma states that $\Pi(w)$ and $\overline{\Pi}(w)$ are equivalent problems.

Lemma 13. *Let $w \in \mathcal{W} \cap \{0,1\}$. Let $\Pi(w)$ be the optimization problem defined as*

$$\min_{D\in\mathcal{L}(w)\cap\{0,1\}} \left(c^T D\right),$$

and $\overline{\Pi}(w)$ *the optimization problem defined as*

$$\min_{\overline{D}\in\overline{\mathcal{L}}(w)\cap\{0,1\}} \left(\overline{c}^T\overline{D}\right),$$

then $\Pi(w)$ *and* $\overline{\Pi}(w)$ *are equivalent problems.*

Sketch of the proof. The proof is provided in Appendix and is in three parts. We first show that there are 9 types of implied constraints for $\Pi(w)$. Second, we prove that these implied constraints are sufficient, in the sense that all original constraints of $\Pi(w)$ can be formulated using linear combinations of implied constraints. Third, implied constraints fix a subset of variables to 0 or 1. Therefore, these variables can be deleted from the formulation since they are constants for this problem. In addition, we reduce the final number of variables by using another implied constraint, thus obtaining $\overline{\Pi}(w)$. □

The following formula provides the transformation between a solution of $\overline{\Pi}(w)$ and $\Pi(w)$ (variables not mentioned in the formula are equal to 0):

$$\begin{aligned}
&d_{\sigma_1} = 1, \\
&f_{r\widetilde{z}_r} = 1, && \forall\ r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B, \\
&d^{(E)}_{r\sigma_k k} = \overline{d}_{r,k-1}, && \forall\ k \in \{2,\dots,\overline{N}\},\ \forall\ r \in \overline{E}_{\nu_{k-1}}, \\
&d^{(L)}_{\sigma_k r k} = \overline{d}_{rk}, && \forall\ k \in \{1,\dots,\overline{N}\},\ \forall\ r \in \overline{E}_{\nu_k}, \\
&f_{rz} = \overline{f}_{rz}, && \forall\ r \in \overline{\mathcal{S}}_B,\ \forall\ z \in \{\widetilde{z}_r,\dots,Z\}
\end{aligned} \tag{5.27}$$

Integrality of $\overline{\Pi}(w)$

The next two main theorems show that,

- any extreme point $D^* = (d^*, f^*)$ of $\overline{\mathcal{L}}(w)$ is such that $d^* \in \{0,1\}$.

- if this extreme point is optimal and γ verifies Condition (A), then $f^* \in \{0,1\}$.

Theorem 3. *Let $w \in \mathcal{W} \cap \{0,1\}$ and $D^* = (d^*, f^*)$ be an extreme point of $\overline{\mathcal{L}}(w)$, then*

$$d^* \in \{0,1\}.$$

Sketch of the proof. The proof is provided in Appendix. We suppose by contradiction that $d^* \notin \{0,1\}$, thus there exists p, l such that $d^*_{pl} \notin \{0,1\}$. We show that there exists q such that $d^*_{ql} \notin \{0,1\}$ and two main cases arise. In each case, we construct D^1, D^2 such that $D^1, D^2 \in \overline{\mathcal{L}}(w)$, $D^1 \neq D^2 \neq D^*$ and $D^* = \frac{1}{2}\left(D^1 + D^2\right)$ which provides a contradiction to D^* being an extreme point. The two cases are:

1. If the request performed during crane cycle l is a retrieval, then p and q are I/O points and we can easily construct D^1 and D^2.

2. If the request performed during crane cycle l is not a retrieval, then constructing D^1 and D^2 is not straightforward and requires the proof of a technical lemma (see Lemma 14 in Appendix).

□

Theorem 4. *Let $w \in \mathcal{W} \cap \{0,1\}$. If D^* is an extreme point of $\overline{\mathcal{L}}(w)$ such that $D^* = \underset{\overline{D} \in \overline{\mathcal{L}}(w)}{argmin} \left(\overline{c}^T \overline{D}\right)$ and γ verifies Condition (A), then*

$$D^* \in \{0,1\}.$$

Consequently, since $\overline{\Pi}(w)$ and $\Pi(w)$ are equivalent problems, then $\Pi(w)$ can be solved by solving the linear program $\Pi^L(w)$ defined by

$$\left\{ \begin{array}{l} \min\limits_{0 \leqslant (\overline{d},\overline{f}) \leqslant 1} \left(\sum\limits_{\substack{k \in \{1,\ldots,\overline{N}\} \\ r \in \overline{E}_{\nu_k}}} \overline{t}_{rk} \overline{d}_{rk} + \sum\limits_{\substack{r \in \overline{\mathcal{S}}_B \\ z \in \{\widetilde{z}_r,\ldots,Z\}}} \beta_z \overline{f}_{rz} \right) \\ \\ s.t. \left\{ \begin{array}{ll} \forall\ k \in \{1,\ldots,\overline{N}\}, & \sum\limits_{r \in \overline{E}_{\nu_k}} \overline{d}_{rk} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B, & \sum\limits_{z \in \{\widetilde{z}_r,\ldots,Z\}} \overline{f}_{rz} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B, & \sum\limits_{z \in \{\widetilde{z}_r,\ldots,Z\}} z \overline{f}_{rz} - \sum\limits_{k \in \overline{K}_r} \overline{d}_{rk} = \widetilde{z}_r, \end{array} \right. \end{array} \right. \tag{5.28}$$

In conclusion, this section provides the first mathematical formulation to solve efficiently the scheduling of retrieval and storage requests while routing the crane and taking into account relocations and storage locations assignments. This mathematical formulation has two types of variables w and D. w corresponds to the scheduling of the requests as well as fixing the loading stacks for storage requests, while D indicates the routing of the crane. The dimension of w is of the order to $\overline{N}^2 M$ (typically hundreds or thousands) while D has a dimension of $(XY)^2\,\overline{N}$ (typically hundreds of thousands or millions).

In this section, we have shown that given a binary vector $w \in \mathcal{W} \cap \{0,1\}$, we can relax the integrality constraints on D and still get a integer solution by solving the linear program $\Pi^L(w)$. Consider $g(w)$ to be the optimal objective function of $\Pi^L(w)$, then our initial problem can be formulated as

$$\min_{w \in \mathcal{W} \cap \{0,1\}} \left(g(w) \right). \tag{5.29}$$

Therefore, since $g(w)$ can be evaluated in polynomial time, solving our problem can be reduced to designing an efficient search algorithm on $\mathcal{W} \cap \{0,1\}$. Since the dimension of $\mathcal{W}$ is relatively much lower than the original feasible space $\mathcal{P}$, we show in the next

section how these results helps solving efficiently the original problem. In addition, this new approach suggests a simple way to solve larger instances where, in the given time limit, the IP could potentially only provide solutions with high cost or not even get a feasible solution. Instead, one could use any meta heuristic search (simulated annealing, genetic algorithms, tabu search, etc...) on the space $w \in \mathcal{W} \cap \{0,1\}$. These heuristics are expected to work better under this new approach than if these were directly implemented for the original problem again due to the relatively small dimension of $\mathcal{W}$.

Instead of investigating which common meta heuristic would work the best, we provide a "simple" local search heuristic on $\mathcal{W} \cap \{0,1\}$ that performs well on this problem and which provides some intuition as well.

Case where γ does not verify condition (A). Theorem 3 shows that any extreme point $D^* = (d^*, f^*)$ of $\overline{\mathcal{L}}(w)$ is such that $d^* \in \{0,1\}$. Simple counter-examples show that we could have $f^* \notin \{0,1\}$. Nevertheless, variables f are only used to compute the cost of a solution while actual decisions correspond to variables d^*. Thus, given $w \in \mathcal{V} \cap \{0,1\}$, solving $\Pi^L(w)$ provides a feasible sequence of decisions as $d^* \in \{0,1\}$. Therefore, the heuristic provided in the next section, which tries to solve the problem defined in Equation (5.29), can still be applied in real operations even in the case where γ does not verify Condition (A). The difference with the original problem is that the part of the cost that involves the variables f^* could be underestimated as we relax the integrality of f^*.

5.4 Heuristic Procedure for Real-Time Operations

Based on the analysis of the previous section, we now design an efficient heuristic method to search the space $\mathcal{W} \cap \{0,1\}$. This section describes a randomized algorithm which decomposes its search into two stages. The first stage takes an order of requests as input and looks for a "good" set of starting stacks by sampling from a smaller promising set of stacks. The second stage builds upon the first stage and searches

in the space of request orders by using a repeated-random-start local search. This algorithm requires two integer inputs $R_1, R_2 \in \mathbb{N}$ respectively corresponding to the number of samples in the first stage and the number of re-starts for the second stage.

5.4.1 Search Space Decomposition

First, we explain the reason to decompose our search strategy in two stages. Notice that constraints defining $\mathcal{W}$ only involve sums of w_{nsk} over $s \in L_n$. This motivates the definition of the polyhedron $\mathcal{V}$. We say that $v \in \mathcal{V}$ if $v = (v_{nk})_{n,k \in \{1,\dots,\overline{N}\}}$ and v verifies the following constraints:

$$\forall\ n \in \{1, \dots, \overline{N}\}, \quad \sum_{k \in \{1,\dots,\overline{N}\}} v_{nk} \ = 1,$$

$$\forall\ k \in \{1, \dots, \overline{N}\}, \quad \sum_{n \in \{1,\dots,\overline{N}\}} v_{nk} \ = 1.$$

$$\forall\ n \in \{n' \in \mathcal{N}_r \cup \mathcal{N}_u \mid b_{n'} \neq 0\},\ \forall\ k \in \{1, \dots, \overline{N}\}, v_{nk} - \sum_{k' \in \{1,\dots,k-1\}} v_{b_n k'} \ \leqslant 0, \quad (5.30)$$

$$\forall\ n \in \{1, \dots, N\},\ \forall\ k \in \{n + \delta_n^+ + 1, \dots, \overline{N}\},$$
$$\sum_{\substack{n' \in \{1,\dots,N\} \\ k' \in \{1,\dots,k-1\}}} v_{n'k'} + \left(k - \left(n + \delta_n^+\right)\right) v_{nk} \ \leqslant k - 1,$$

$$\forall\ n \in \{1, \dots, N\},\ \forall\ k \in \{1, \dots, \overline{N} - (N - n + \delta_n^-) - 1\},$$
$$\sum_{\substack{n' \in \{1,\dots,N\} \\ k' \in \{k+1,\dots,\overline{N}\}}} v_{n'k'} + \left(\overline{N} - k - \left(N - n + \delta_n^-\right)\right) v_{nk} \ \leqslant \overline{N} - k.$$

Note that if $v \in \mathcal{V}$, then for any w such that $v_{nk} = \sum_{s \in L_n} w_{nsk}$ we have $w \in \mathcal{W}$. Therefore, we can reformulate the problem under the point of view of Equation (5.29) into

$$\min_{v \in \mathcal{V} \cap \{0,1\}} \left(\min_{\substack{w \in \{0,1\} \\ v_{nk} = \sum_{s \in L_n} w_{nsk}}} (g(w)) \right),$$

which itself can be written as

$$\min_{v \in \mathcal{V} \cap \{0,1\}} \left(\min_{\sigma \in L} \left(h(v, \sigma) \right) \right). \tag{5.31}$$

where $L = \bigotimes_{n \in \{1,\ldots,\overline{N}\}} L_n$ and $h(v, \sigma) = g(w)$, such that $w_{nsk} = v_{nk} \mathbb{1}\{s = \sigma_n\}$. In this last formulation, it is important to notice that by definition $L_n = \{s_n\}$ for $n \in \mathcal{N}_r \cup \mathcal{N}_u$, hence σ_n is fixed, thus only σ_n for $n \in \mathcal{N}_s$ are actual variables. In conclusion, the problem formulated in Equation (5.31) should be seen as a two-step process. First, given $v \in \mathcal{V} \cap \{0, 1\}$, the goal is to find σ where σ_n corresponds to the stack where the loaded crane drive performing request n starts that minimizes $h(v, \sigma)$. In a second step, the goal is to find the best $v \in \mathcal{V} \cap \{0, 1\}$. The main reason to decompose the problem in this way is that σ does not have any coupling constraints while v is constrained in several ways (assignment, flexibility and precedence constraints). Therefore we suggest to use two different search strategies for these two types of variables.

5.4.2 First Stage: Restricted Sampling on L

In this section, we consider $v \in \mathcal{V} \cap \{0, 1\}$ and we define

$$\kappa_n = \left\{ k \in \{1, \ldots, \overline{N} \mid v_{nk} = 1 \right\}, \ \forall\, n \in \{1, \ldots, \overline{N}\}.$$

As we mentioned, the goal is to be able to compute efficiently the function

$$\phi(v) = \min_{\sigma \in L} \left(h(v, \sigma) \right).$$

Note that a greedy evaluation requires an exponential number of evaluations of the function $h(v, .)$ as $|L| = \prod_{n \in \mathcal{N}_s} |L_n|$. Instead, we approximate $\phi(v)$ by sampling several promising σ and retain the best solution. The idea that we propose is to take advantage of the LP relaxation of the binary integer program of Section 5.3 where we fix the order of requests according to v. More specifically, consider the linear program

$\Lambda(v)$:

$$\min_{(w,D)} \left(c^T D\right) \qquad \text{Equation (5.20)}$$

$$s.t. \begin{cases} D \in \mathcal{D} & \text{Equations (5.7)-(5.12)} \\ (w, D) \in \mathcal{L} & \text{Equations (5.13)-(5.14)} \\ \sum_{s \in L_n} w_{ns\kappa_n} = 1 & \forall\ n \in \{1, \ldots, \overline{N}\} \end{cases}$$

where the last constraints insure that the order defined by v is respected. Note that this formulation can be sped up by setting some variables to zero (details are provided in the Appendix). Let $\left(w^\Lambda, D^\Lambda\right)$ be an optimal solution of this linear program, then we define $L\left(v\right)$ such that

$$L\left(v\right) = \left\{\sigma = (\sigma_1, \ldots, \sigma_{\overline{N}}) \in L \;\middle|\; w^\Lambda_{n\sigma_n\kappa_n} > 0,\ \forall\ n \in \{1, \ldots, \overline{N}\}\right\}. \tag{5.32}$$

We have noticed in the experiments that $|L\left(v\right)|$ is relatively small compared to $|L|$. Moreover, the selected stacks are indeed promising as they have been selected based on the LP relaxation. The other advantage of this procedure is that $\lambda(v) = c^T D^\Lambda$ provides a lower bound on the best attainable cost when the order of requests is set by v, which we will use in the 2^{nd} stage search. Given $L\left(v\right)$ and w^Λ, we sample $\sigma = (\sigma_1, \ldots, \sigma_{\overline{N}})$ using w^Λ as weights such that we have

$$\forall\ n \in \{1, \ldots, \overline{N}\}, \mathbb{P}\left[\sigma_n = s\right] = w^\Lambda_{ns\kappa_n} \ \text{ and } \ \mathbb{P}\left[\sigma = (s_1, \ldots, s_{\overline{N}})\right] = \prod_n \mathbb{P}\left[\sigma_n = s_n\right]. \tag{5.33}$$

Note that, thanks to the last constraint of $\Lambda(v)$, this is a well defined probability distribution. In conclusion, given a certain $v \in \mathcal{V} \cap \{0, 1\}$, we can

1. Solve $\Lambda(v)$ to get w^Λ and $\lambda(v) = c^T D^\Lambda$.

2. Use w^Λ to define $L(v)$ from Equation (5.32) and a probability distribution on this subset of L from Equation (5.33).

3. Sample without replacement $R'_1 = \min\left\{R_1, |L\left(v'\right)|\right\}$ points from the afore-

mentioned probability distribution over $L(v)$ (denoted by $\left(\sigma^1, \ldots, \sigma^{R'_1}\right)$) and compute

$$\psi(v) = \min_{r \in \{1,\ldots,R'_1\}} \left(h(v, \sigma^r)\right). \tag{5.34}$$

First note that $\psi(v) \geqslant \phi(v)$ almost surely. The purpose of $\psi(v)$ is to provide a good randomized approximation of $\phi(v)$. As $R_1 \to |L(v)|$, then $\psi(v) \to \min_{\sigma \in L(v)} (h(v, \sigma))$ almost surely. Thanks to the way $L(v)$ is constructed, we empirically observe this latter value to be close to $\phi(v)$.

5.4.3 Second Stage: Repeated-Random-Start Local Search on $\mathcal{V} \cap \{0,1\}$

This algorithm is an adaptation of a classical local search algorithm which repeatedly starts from a random feasible solution and improves this solution until a local minimum is reached. This algorithm is parameterized by R_2, the number of repeated random starts. Its output is the best local minimum that was found among the R_2 explored ones. In this framework, we use the 1^{st} stage procedure and an algorithm for sampling on $\mathcal{V} \cap \{0,1\}$ that we describe subsequently. The pseudocode of this procedure is provided in Algorithm 4. Note that the definition of neighborhood is given in the pseudocode of Algorithm 4 (lines 10-12). Let $v(i)$ be the current solution of the local search. We consider two requests n and m such that $n < m$ (line 10). We then consider v' such that the crane cycles of n and m are exchanged between $v(i)$ and v' and all other requests are performed during the same crane cycles (line 11). This automatically implies that $v' \in \{0,1\}$. If the precedence constraints are verified i.e. $v' \in \mathcal{V}$, then v' is a neighbor of $v(i)$ (line 12). We also mention that lines 13-14 are added to enhance the speed of the local search. Indeed, when solving $\Lambda(v')$, we have access to $\lambda(v') \leqslant \psi(v')$. If $\lambda(v') \geqslant \psi(v(i))$, then we know that $\psi(v') \geqslant \psi(v(i))$, hence no need to sample for on $L(v')$ as v' cannot improve the current solution.

Sampling in $\mathcal{V} \cap \{0,1\}$, an Accept-and-reject approach. We now describe the procedure SAMPLE_$\mathcal{V}()$. Note that a point $v \in \mathcal{V} \cap \{0,1\}$ corresponds to a complete

Algorithm 4 Heuristic based on Repeated-Random-Search Algorithm.

procedure $\left(v^{RRS}, \sigma^{RRS}\right)$ = REPEATED_RANDOM_SEARCH $(R_1,\ R_2)$
  for $i = 1, \ldots, R_2$ **do**
    Compute $v(i)$ = SAMPLE_$\mathcal{V}()$;
    Solve $\Lambda(v(i))$. Get $L\left(v(i)\right)$ from Equation (5.32) and $R_1' = \min\{R_1, |L\left(v(i)\right)|\}$;
    for $r = 1 \ldots, R_1'$ **do** Sample σ^r without replacement from Equation (5.33);
    Compute $\psi\left(v(i)\right) = \min\limits_{r \in \{1,\ldots,R_1'\}} \{h(v(i), \sigma^r)\}$ and $\sigma(i) = \operatorname{argmin}\limits_{r \in \{1,\ldots,R_1'\}} \{h(v(i), \sigma^r)\}$;
    $j = 0$;
    while $j < \overline{N}\left(\overline{N} - 1\right)/2$ **do**
      Increment $j = j + 1$;
      Sample without replacement $(n, m) \in \{1, \ldots, \overline{N}\}^2$ s.t. $n < m$;
      Consider v' such that $\kappa_n' = \kappa_m(i)$, $\kappa_m' = \kappa_n(i)$ and $\kappa_p' = \kappa_p(i)$, $\forall\ p \neq n, m$;
      if $v' \in \mathcal{V}$ **then**
        Solve $\Lambda(v')$ to get $\lambda\left(v'\right)$;
        if $\lambda\left(v'\right) < \psi\left(v(i)\right)$ **then**
          Compute $L\left(v'\right)$ from Equation (5.32). Let $R_1' = \min\{R_1, |L\left(v'\right)|\}$;
          for $r = 1 \ldots, R_1'$ **do** Sample σ^r without repl. from Equation (5.33);
          Get $\psi\left(v'\right) = \min\limits_{r \in \{1,\ldots,R_1'\}} \left\{h(v', \sigma^r)\right\}$ and $\sigma' = \operatorname{argmin}\limits_{r \in \{1,\ldots,R_1'\}} \{h(v', \sigma^r)\}$;
          if $\psi\left(v'\right) < \psi\left(v(i)\right)$ **then** $v(i) = v'$, $\sigma(i) = \sigma'$ and $j = 0$;
  return $\left(v^{RRS}, \sigma^{RRS}\right) = \operatorname{argmin}\limits_{i \in \{1,\ldots,R_2\}} \{h(v(i), \sigma(i))\}$;

matching between requests and crane drives. Based on studies to sample efficiently on complete matching (e.g., in [34]), we use the common Accept-and-reject approach to sample random points in $\mathcal{V} \cap \{0, 1\}$. The main difference with typical studies is that in addition to having matching constraints, there are precedence constraints to take into account (see Equation (5.30)). The pseudocode of SAMPLE_$\mathcal{V}()$ is given in Algorithm 5.

The idea behind this sampling algorithm is simple: For each crane cycle, assign randomly a request that can be assigned to this crane cycle given the flexibility requirements and the precedence constraints. If, no request can be assigned to a given crane drive, then restart the process until a complete matching satisfying all flexibility and precedence constraints is found.

Algorithm 5 Sampling algorithm using Accept-and-reject.

procedure (v) = SAMPLE_$\mathcal{V}()$
  $v = \{0\}^{\overline{N}^2}$, $U(1) = \{1, \ldots, \overline{N}\}$, $k = 0$ and $k^p = 1$;
  while $k < \overline{N}$ **do** Increment $k = k + 1$;
    $\mathcal{R}(k^p, k) = U(k) \cap (\{n \in \mathcal{N}_s \cup \mathcal{N}_r \mid n - \delta_n^- \leqslant k^p \leqslant n + \delta_n^+\} \cup \mathcal{N}_u)$;
    while $|\mathcal{R}(k^p, k)| > 0$ **do** sample n uniformly in $\mathcal{R}(k^p, k)$. Take $v' = v$ and $v'_{nk} = 1$
      if v' satisfies precedence constraints in Equation (5.30) $\forall\, k' \in \{1, \ldots, k\}$ **then**
        $v = v'$, $U(k+1) = U(k) \setminus \{n\}$ and **break**;
      $\mathcal{R}(k^p, k) = \mathcal{R}(k^p, k) \setminus \{n\}$;
    if $|\mathcal{R}(k^p, k)| = 0$ **then** $v = \{0\}^{\overline{N}^2}$, $U(1) = \{1, \ldots, \overline{N}\}$, $k = 0$ and $k^p = 1$;
    else if $n \in \mathcal{N}_s \cup \mathcal{N}_r$ **then** Increment $k^p = k^p + 1$;
  return v;

5.5 Computational Experiments

In this section, we first compare the efficiency of the different methods developed in the previous sections through randomly generated instances. Afterwards, we use real data from a real terminal to show the potential gain of using our heuristic method compared to the actual practice. The study is performed on one processor (2.6 GHz Intel E5-2690 v4) of a Dell C6300 with 4 gigabytes of RAM. The programming language is Julia 0.5.0 and all optimization problems are solved using Gurobi 7.0.1.

Important Note. In all the following experiments, *all algorithms integrate the practical constraint referred to as "restricted" in the CRP literature* (see assumption $A1$ in [10]). This constraint requires each retrieval request n to be directly preceded by the relocation requests needed to retrieve container n. Mathematically, it can be formulated as:

$$\forall\, n \in \{n' \in \mathcal{N}_r \cup \mathcal{N}_u \mid b_{n'} \in \mathcal{N}_u\}, \forall\, k \in \left\{2, \ldots, \overline{N}\right\},\ v_{nk} = v_{b_n, k-1}.$$

We show that even under this practical constraint, our algorithms have a major impact on operations. Future work could include more experiments without this additional constraint.

5.5.1 Randomly Generated Instances

Using 30 randomly generated instances as explained below, we first assess the performance of the IP based algorithm and our heuristic (as a function of γ) compared to a baseline. Subsequently, we evaluate the impact of the parameters δ and N on the performance of the heuristic. In all experiments, the performance indicator is the average travel cost of the crane per productive request (i.e., the total cost to perform all requests divided by the total number of productive requests).

Simulation parameters

Each instance is defined by an initial block configuration, a sequence of productive requests and a sequence of number of requests known in advance (i.e., a sequence of Ns used in each optimization problem). Note the sequence of Ns must sum up to the total number of requests.

To generate an initial block configuration, we consider that the block has dimensions $X = 7$ rows, $Y = 30$ bays and $Z = 4$ tiers, giving a total of $XY = 210$ stacks. The I-O point configuration is the *right-sided Asian configuration*. Moreover, we assume that the initial number of containers is equal to $\lfloor 0.67\,(XYZ - Y(Z-1))\rfloor = 502$ where 0.67 is called the fill rate. Finally, the position of each container is drawn uniformly at random.

Each sequence of requests consists of 1500 productive requests that we generate randomly one at a time. A new productive request is equally likely to be a storage request or a retrieval request, given the fact that the number of containers denoted by C must satisfy $XY \leqslant C \leqslant XYZ - Y(Z-1)$ at all times. If the new request is a retrieval request, we assume that the container to be retrieved is picked at random among the ones that have spent at least a certain number of requests in the block. We set this number of requests to 210 (i.e., a container can be retrieved only if there are at least 210 productive requests between its arrival and its departure from the block). In addition, we assume that each container initially present in the block can be retrieved as soon as the first retrieval request.

Concerning the sequence of number of requests known in advance, we consider, unless specified otherwise, that N remains constant and is taken to be $N = 5$. To solve each optimization problem, the actual arrival order of trucks is required as an input. Because the time at which this arrival order is known is clearly limited, it is not really realistic to consider a much larger N (see the data processing in the next section).

In these experiments, we consider a block for import containers, i.e., storage requests are carried out by internal trucks and retrieval requests by external trucks. A flexibility policy of interest for port operators is $(0, \delta)$ for internal trucks and $(\delta, 0)$ for external trucks, where $0 \leqslant \delta \leqslant N$. In practice, this corresponds to enforcing external customers to be served at least before their position and internal trucks not too much after their position in order not to delay ships significantly. Unless specified otherwise, we consider

$$\delta = \left\lfloor \frac{N}{2} \right\rfloor = 2,$$

i.e., retrieval requests have a flexibility of $(2, 0)$ while storage requests have a flexibility of $(0, 2)$.

Each algorithm is given the realistic time limit of *60 seconds* to solve one optimization problem. Other parameters of the problems (crane speeds,...) are provided in Appendix C.1. Results for the heuristic method are reported for $(R_1, R_2) = (40, 40)$.

Performance of different algorithms

We compare the following algorithms:

- **Baseline**, the binary integer program introduced in section 5.3 with $\gamma = 0$ and $\delta = 0$, which corresponds to the greedy optimization of the routing of the crane without considering future relocations, under the FCFS constraint for the order of requests.

- **Heuristic**, as described in the previous section for different values of γ and $\delta = 2$.

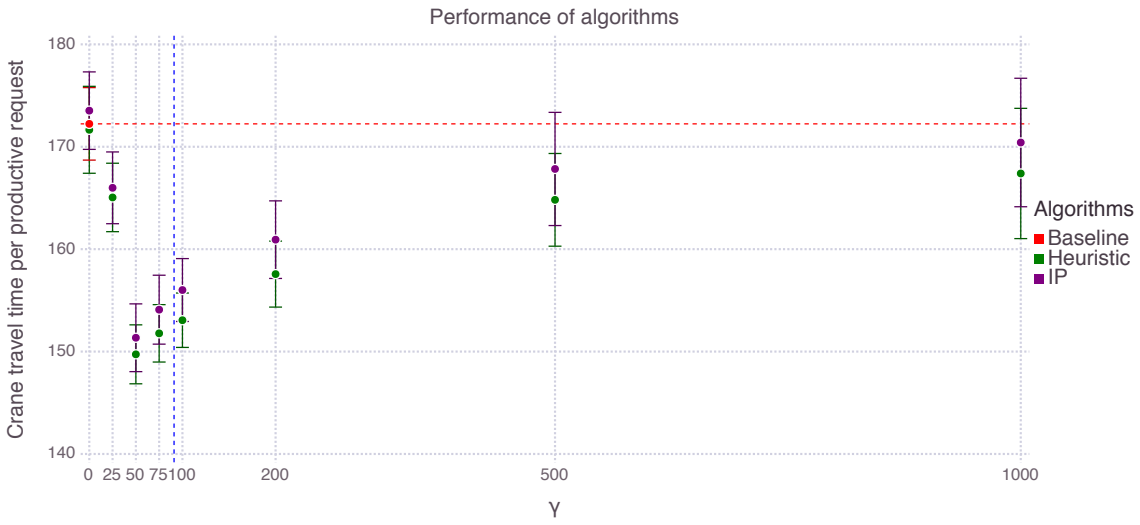


Figure 5-4: Performance of algorithms as function of γ: Each point represents the mean indicator obtained by different algorithms over the 30 randomly generated instances and the error bars represent ± 1.645 standard deviations. The red horizontal line corresponds to the mean of the baseline and the blue vertical line correspond to the lower bound on γ in Condition (A).

- **IP**, the binary integer program introduced in section 5.3 for different values of γ and $\delta = 2$.

We tested $\gamma \in \{0, 25, 50, 75, 100, 200, 500, 1000\}$. We can draw several insights from Figure 5-4:

1. There appears to be an optimal γ and $\gamma = 50$ is the best observed value for both the heuristic and the IP in the setting of our experiments. Intuitively, increasing the weight on the cost-to-go improves significantly the solution (up to 13% of improvement). However, by putting too much weight on the cost-to-go, both the heuristic and the IP worsen as they neglect the immediate cost. **From this point forward, we consider the heuristic with $\gamma = 50$.** As a side note, the best γ (50) in this particular setting does not verify condition (A).

2. The heuristic is performing better on average than the IP given the practical time limit of 60 seconds to solve every optimization problem (even though this difference is not statistically significant). This demonstrates the value of using the heuristic over the IP even when $N = 5$. Table 5.2 reports the percentage

of optimization problems that are not proven to be solved optimally by the IP, averaged over all 30 instances. There is a significant number of such cases, explaining the difference between the IP and the heuristic.

γ	0	25	50	75	100	200	500	1000
% of problems not proven to be solved optimally by the IP	35%	25%	36%	47%	43%	47%	49%	48%

Table 5.2: Percentage of optimization problems not proven to be solved optimally by the IP in the practical time limit of 60 seconds.

Impact of the parameter δ

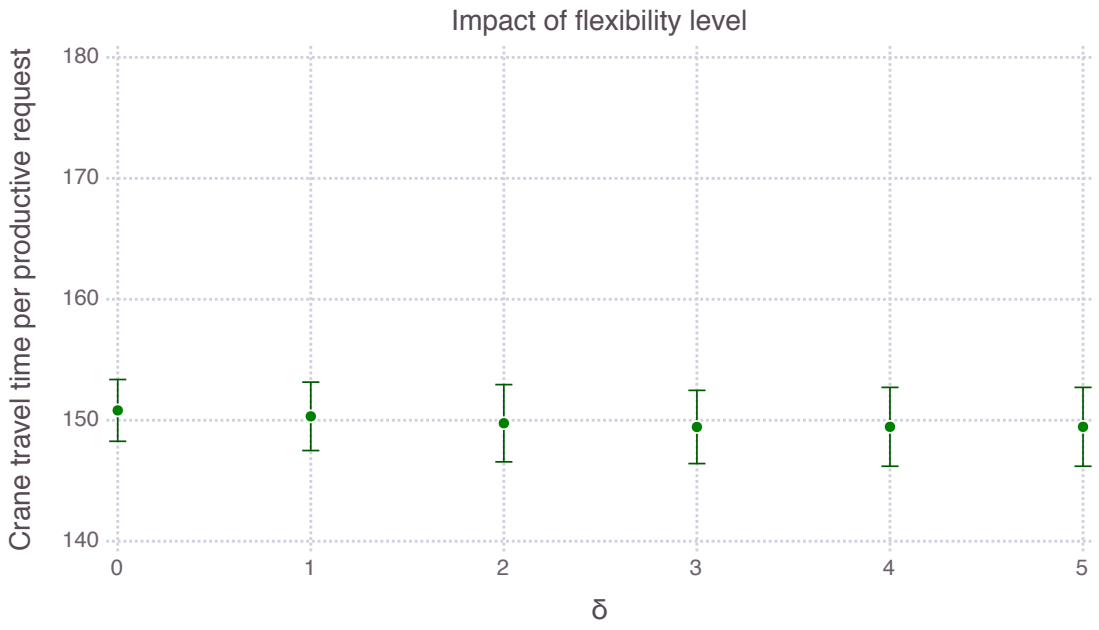


Figure 5-5: Impact of δ: Each point represents the mean indicator obtained by the heuristic with $\gamma = 50$ over all 30 instances and the error bars represent ± 1.645 standard deviations.

We now study the impact on the heuristic solution of different levels of flexibility by varying δ and keeping $N = 5$. Based on the results presented in Figure 5-5, the following insights can be obtained:

1. Increasing flexibility in the order of requests has a positive impact on the average crane travel time (about 1%) but this benefit is not statistically significant and is relatively small compared to the benefit of a well tuned γ (see Figure 5-4). As the flexibility level solely depends on the port operator's policy, this latter can be set to balance the crane's efficiency (quantified in Figure 5-5) and truck driver's tolerance to this flexibility.

2. Most of this benefit is captured by setting a flexibility of $\delta = 2$ or 3. This relates to the general intuition that in scheduling problems, most of the benefits of flexibility is captured when δ is around half of N. Increasing δ might help in some cases but on average a flexibility of $N/2$ is close to achieve the benefits of a fully flexible system.

Impact of the parameter N

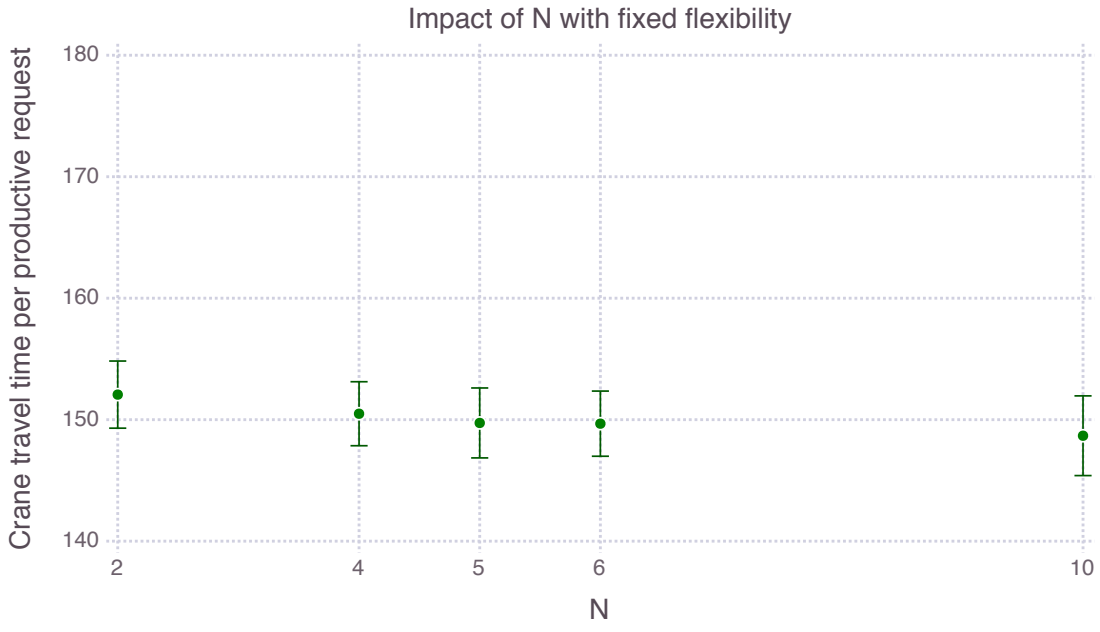


Figure 5-6: Impact of N: Each point represents the mean indicator obtained by the heuristic with $\gamma = 50$ over all 30 instances and the error bars represent ± 1.645 standard deviations.

In some port terminals, the mean arrival rate of productive requests might be

different which would lead to a different N. Recall that, due to practical constraints (the full arrival order of trucks must be known), thus N cannot be arbitrarily large. Consequently, this experiment considers $N \in \{2, 4, 5, 6, 10\}$ while keeping a flexibility of $\delta = 2$. Figure 5-6 quantifies the benefit of having a larger N (around 1-2%). Indeed, having more information increases the impact that the heuristic can have on efficiency. But, this positive impact appears again relatively small when put in perspective with Figure 5-4. We also performed the experiment where $\delta = \lfloor N/2 \rfloor$ (varies with N). We observe very similar results with N having a slightly bigger impact (on the order of 2-3%). Most importantly, note that γ is kept constant, so a bigger impact of N might be observed by varying γ for each N.

5.5.2 Data from a Real Terminal

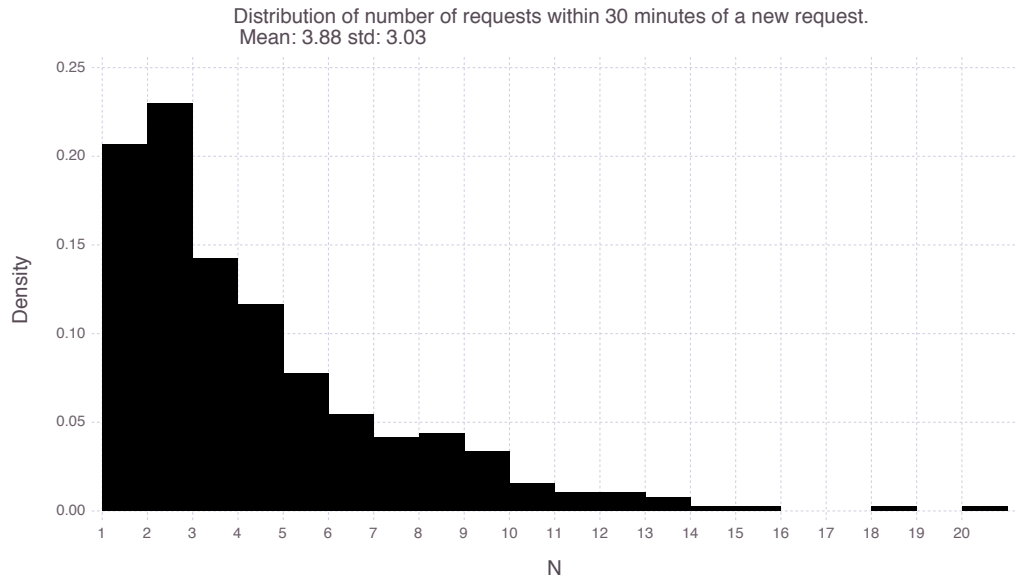
Data processing

We collected data from two import blocks of a real port for 17 days in September 2017. The data included the position of each container in these two blocks on 09/07/2017 at 05:35:04 AM local time. For the next 17 days, each move of the cranes operating in these blocks was recorded. We summarize the main figures of this dataset in Table 5.3. From this data, we can extract the initial block configuration as well as the sequence of productive requests. Recall that the third piece of data needed for our

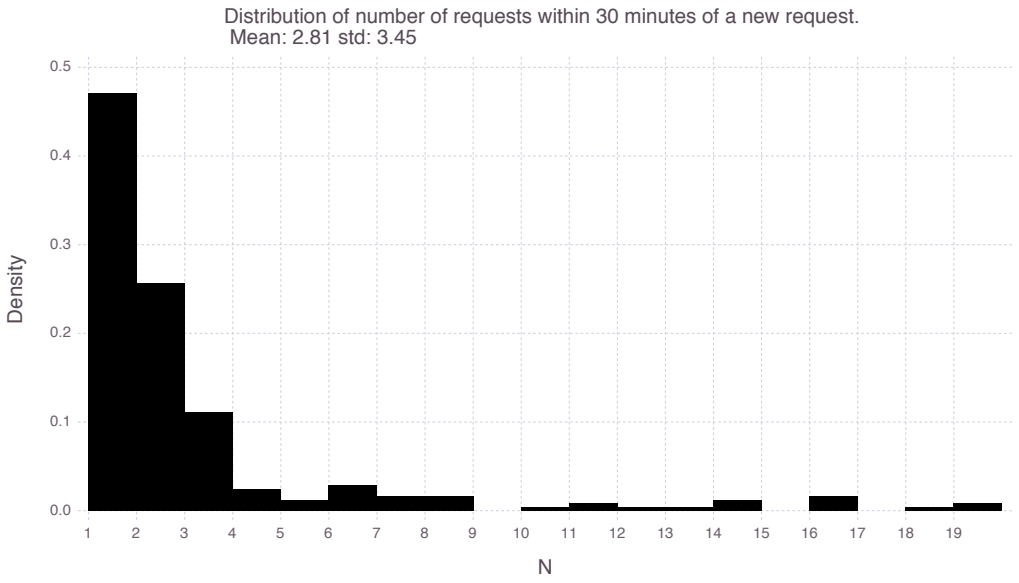
Parameter	X	Y	Z	C	IO-Points	# of productive moves (requests)	# of unproductive moves (relocations)
Block 1	7	19	5	297	right-sided Asian	1502	729
Block 2	7	20	5	185	right-sided Asian	679	201

Table 5.3: Data summary for requests in two blocks for 17 days in 9/2017.

simulation is the sequence of Ns. To infer this from our data, we make the reasonable assumption that each request is available a fixed amount of time before the request was actually performed. The operator suggested we fix this amount of time to *30 minutes*. Because we have access to every time-stamp for all requests, we can construct the sequence of Ns. More precisely, for each new request (i.e., not yet considered in an



5-7a Block 1.



5-7b Block 2.

Figure 5-7: Distribution of N of requests from two blocks for 17 days in September 2017.

optimization problem), a new problem is considered with N equal to the number of requests available in the 30 minutes following the new request. Because this is close to what actually happens, we refer to this sequence of Ns as the *real scenario*. Figure

5-7 presents the distributions of N for the two blocks over the 17 days. From these distributions, we also consider the *ideal scenario* where N remains constant and is taken to be the rounded mean of the aforementioned distributions ($N = 4$ for Block 1 and $N = 3$ for Block 2). Note that to consider the same number of productive requests, the last optimization problem may use a different N than the mean.

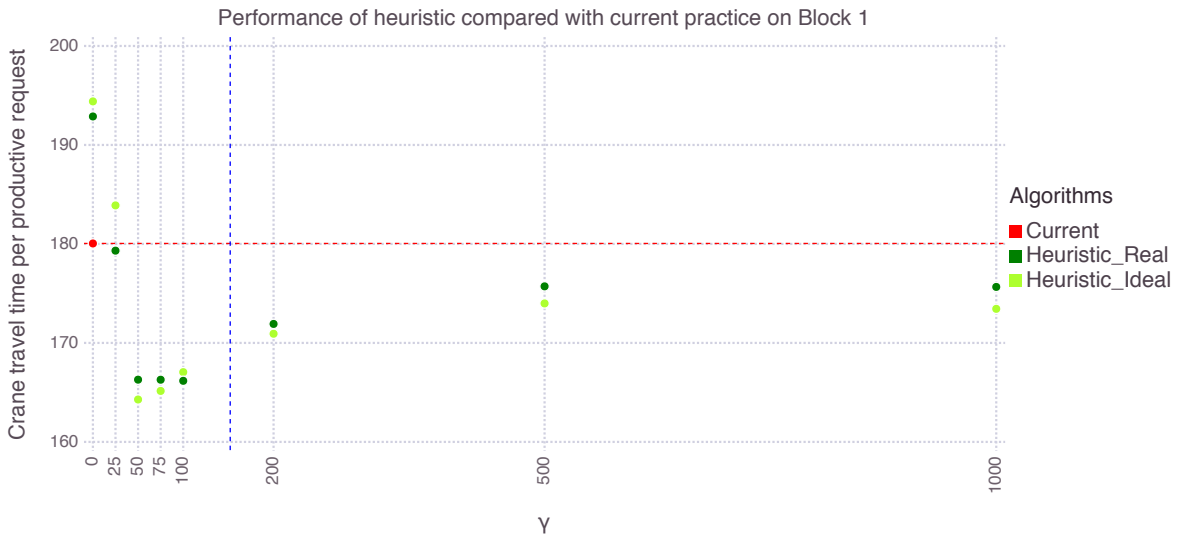
Finally, we consider the same parameters as for randomly generated instances: retrieval requests have a flexibility of $(2, 0)$ while storage requests have a flexibility of $(0, 2)$; the heuristic uses $(R_1, R_2) = (40, 40)$ and is given 60 seconds to solve each optimization problem; other speed parameters are given in Appendix C.1.

Results

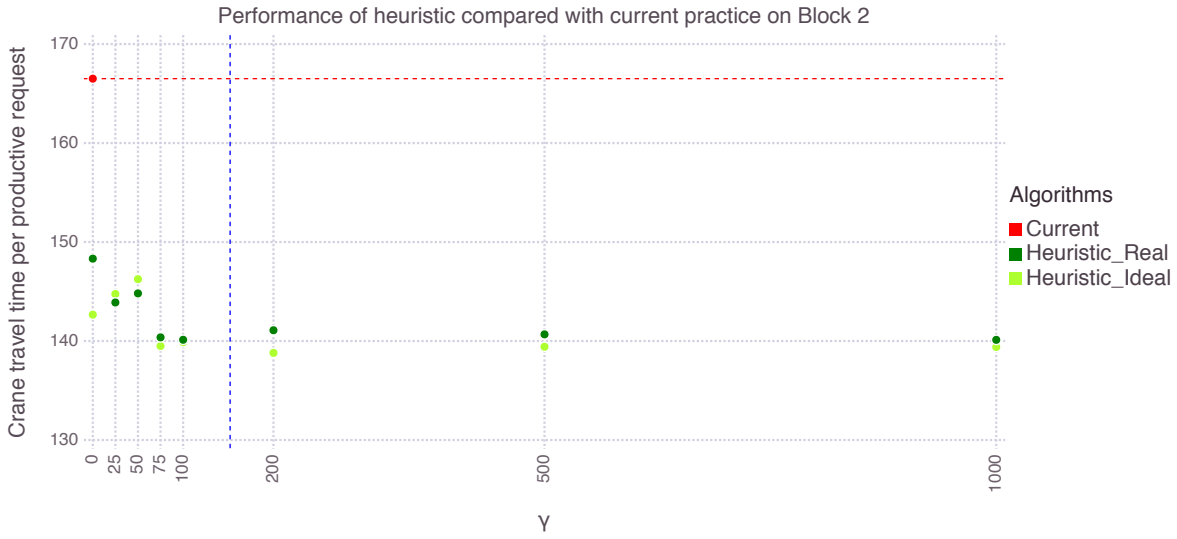
Results are reported in Figure 5-8 for both blocks. These experiments provide three main insights:

1. Most importantly, using a good value for γ (here, 100) leads to a significant improvement over the current practice. In the real scenario, the improvement is of the order of 8% in Block 1 and 16% in Block 2, therefore proving the efficacy of our proposed method in real operations.

2. The performance of the heuristic as a function of γ in Block 1 are quite similar to the ones from the randomly generated instances, with the noticeable difference that the current practice is clearly outperforming the heuristic for $\gamma = 0$ (thus our artificial baseline which was even worse). This is not surprising as current practice should be taking future relocations into account, thus only a γ properly set can outperform the current practice. In the case of Block 2, the impact of γ appears to be quite different. Indeed, any value of γ outperforms clearly the current practice but larger values of γ are similar to the best value of γ (100). Indeed, the heuristic has a very similar behavior for all $\gamma \geqslant 75$. Intuitively, increasing γ should not change the heuristic after a certain point, because it tends to minimize only future relocations. In this case, this point is relatively small (75) because empty stacks are always quite close to the current position



5-8a Block 1.



5-8b Block 2.

Figure 5-8: Performance of heuristic algorithm as function of γ on real data: Each point represents the average crane travel time obtained by the heuristic under the real and ideal scenarios. The red horizontal line corresponds to the mean of the current practice and the blue vertical line correspond to the lower bound on γ in Condition (A).

of the crane (due to the low fill rate of the block of about 0.3). The difference in the results for $\gamma \geqslant 75$ can mostly be explained by the randomized nature of the heuristic. Therefore, taking future relocations into account seems to

have an even more significant impact by reducing dramatically the number of relocations.

3. Finally, an interesting point is the small difference between the real and ideal scenarios (at most on the order of 3%). This validates our analysis from the randomly generated data where N is taken to be constant and future experiments could make this assumption for real scenarios. In addition, the ideal scenario show a slightly larger improvement than the real scenario for most cases. This suggests that having a steady flow of requests could potentially slightly improve the operations as compared to having rush/empty hours.

Note that the data only represents two particular instances and the conclusions we have drawn here might not apply in other specific cases.

5.5.3 Main Insights

These experiments (both synthetic and from real data) lead us to conclude that the most important parameter of our model is the parameter γ. Variations of this parameter can lead to significant improvement over the baseline and current practice. This parameter tuning appears to be all the more crucial in the case of high fill rates. Once this parameter is fixed, allowing for some flexibility in the order of requests is not expected to provide much if any benefit.

Chapter 6

Concluding Remarks

We conclude this thesis with a summary of contributions, emphasizing our technical contributions, and a discussion of future research directions.

6.1 Summary

In Chapter 1, we discuss the growing importance of container terminals as nodes of worldwide supply chains. We provide background information about equipment and terminal operations. We present operations research models developed to answer the challenges arising from the recent demand for higher productivity of maritime terminals.

In Chapter 2, we review papers addressing the container relocation problem and its variants, as well as the yard crane scheduling problem. We also briefly review the literature for the stacking problem and the pre-marshalling problem.

In Chapter 3, we study the container relocation problem as it was originally introduced in [38]. First, we present a new binary integer program (CRP-I) using an enhanced binary encoding of the CRP. The major contributions of this work are the novelty, the efficiency (in terms of variables and constraints), and the adaptability of this mathematical formulation for the restricted CRP. These features provide a strong basis for future research using this approach. Finally, our formulation outperforms in terms of computation time all previous mathematical formulations and most other

exact methods, except the B&B approach from [66]. In addition, as shown by our formulation and other exact approaches, the container relocation problem is known for its computational intractability, so most research studies have designed heuristics to solve the problem, particularly for large configurations. We show a new theoretical result stating that the ratio between the expected minimum number of relocations and a simple lower bound (given by Lemma 2) approaches 1. The main insight of this result is that in large configurations each blocking container is relocated at most once with high probability. This leads us to believe that the same theoretical result should hold for heuristic MinMax (see [10]) and we confirm this intuition by simulation.

In Chapter 4, we extend the CRP to the more practical case in which the retrieval order of containers is not known far in advance. First, we introduce a new stochastic model, called the batch model, show the applicability of this model and compare it theoretically with the existing model of Zhao and Goodchild [86]. Then, we derive lower bounds and fast and efficient heuristics for the SCRP. Subsequently, we develop two novel algorithms ($PBFS$ and $PBFSA$) to solve the stochastic CRP in different settings. Efficiencies of all algorithms are demonstrated through computational experiments, for which all results are made available online. Finally, using our solution methods and based on extensive experiments, we conjecture the optimality of the simple leveling heuristic in the online stochastic setting. More generally, the methods developed in this chapter apply to multistage stochastic optimization problems, where the number of stages is finite, the set of feasible actions at each stage is finite, the objective function is bounded and bounds on the objective function can be easily computed.

Chapter 5 is the first work that integrates the container relocation problem and the yard crane scheduling problem and makes decisions for storage, retrieval and enforced relocation requests in a realistic setting. Our model jointly optimizes current crane travel time and expected future relocations. First, we formulate this problem as a binary integer programming model. Then, we leverage theoretical properties of this formulation to develop a heuristic based on a reduced state space decomposition and repeated random start local search. Finally, computational experiments on both

data randomly generated and data from a real terminal are conducted to show the efficiency and practicality of our heuristic method compared to current practice. These experiments highlight for practitioners the importance of balancing crane travel time and future unproductive moves (which is shown through the importance of tuning the parameter γ).

6.2 Future Research Directions

6.2.1 Direct Extensions from the Thesis

Section 3.3. Future research could include the use of combinatorial lower bounds in [87, 66] to increase the efficiency of our binary integer formulation for the restricted CRP. Another direction could be the application of other typical techniques such as branch-and-cut or branch-and-price to this formulation. Finally, an important study could investigate the application of ideas provided in this paper to related problems (pre-marshalling, dynamic CRP, etc.).

Section 3.4. A future average case analysis could include the formal proof of a similar result for the heuristic MinMax or the proof of convergence of the difference between the optimal solution and the simple lower bound.

Chapter 4. Future work could include the proof of Conjecture 1, which would have both theoretical and practical impacts. A further study of the CRP under an adversarial model as in [84] could also be of interest.

Chapter 5. Future research could find more efficient solutions for this new model and ways to automatically set γ based on simple business criteria. An important future work could consider our model with multiple crane systems in the case of ports dealing with crane interference (see [63]). Finally, an important work could include information from prediction models or truck appointment systems (such as in Chapter 4).

6.2.2 New Challenges for Storage Yards

Finally, we mention two directions that we believe could be crucial in the next decades for storage yard systems in maritime container terminals:

1. the optimal design of time windows for a truck appointment system (see [82]). As many ports start to implement their own TAS, it is really important to quantify the impact of such a system. On one hand, small time windows imply more information on the retrieval sequence, hence higher operational efficiency of port operators. On the other hand, large time windows insure higher flexibility for truck drivers and a high rate of on-time arrivals. To balance this trade-off, one would need to quantify two important metrics with respect to the expected number of relocations: the "value of information" and the assignment of containers to "wrong" batches.

2. the tactical allocation of incoming containers to blocks at the port level. In Chapter 5, we have assumed that requests are an input to our problem. However, the block to which a container is sent to be stored is a decision for the operator. Future research on this problem could optimize block allocation by modeling each block using our model and use simulation or queuing models to model the whole port terminal.

Bibliography

[1] Dimitri P. Bertsekas. In *Dynamic Programming and Optimal Control, Third Edition*, volume 1 of *Lecture Notes in Computer Science*. Athena Scientific, 2005.

[2] Christian Bierwirth and Frank Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627, 2010.

[3] Joseph Bonney. US ports move toward truck appointment model. *Information Handling Services IHS*, 2015. (Accessed 11.01.2017).

[4] S. Borjian, V. Galle, V. H. Manshadi, C. Barnhart, and P. Jaillet. Container Relocation Problem: Approximation, Asymptotic, and Incomplete Information. *CoRR*, abs/1505.04229, 2015. (Accessed 11.01.2017).

[5] S. Borjian, V. H. Manshadi, C. Barnhart, and P. Jaillet. Managing Relocation and Delay in Container Terminals with Flexible Service Policies. *CoRR*, abs/1503.01535, 2015. (Accessed 11.01.2017).

[6] Andreas Bortfeldt and Florian Forster. A tree search procedure for the container pre-marshalling problem. *European Journal of Operational Research*, 217(3):531–540, 2012.

[7] Héctor J Carlo, Iris FA Vis, and Kees Jan Roodbergen. Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2):412–430, 2014.

[8] Héctor J. Carlo, Iris F.A. Vis, and Kees Jan Roodbergen. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, 236(1):1–13, 2014.

[9] Marco Caserta, Silvia Schwarze, and Stefan Voß. A New Binary Description of the Blocks Relocation Problem and Benefits in a Look Ahead Heuristic. In *Evolutionary Computation in Combinatorial Optimization: 9th European Conference, EvoCOP 2009, Tübingen, Germany, April 15-17, 2009. Proceedings*, pages 37–48. Springer Berlin Heidelberg, 2009.

[10] Marco Caserta, Silvia Schwarze, and Stefan Voß. A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1):96–104, 2012.

[11] Marco Caserta and Stefan Voß. A Corridor Method-Based Algorithm for the Pre-marshalling Problem. In *Applications of Evolutionary Computing: EvoWorkshops 2009*, volume 219, pages 788–797. 2009.

[12] B. Casey and E. Kozan. Optimising container storage processes at multimodal terminals. *Journal of the Operational Research Society*, 63(8):1126–1142, 2012.

[13] Jiang Hang Chen, Der-Horng Lee, and Jin Xin Cao. A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):266–275, 2012.

[14] Philip Davies. Container terminal reservation systems. In *3rd Annual METRANS National Urban Freight Conference, Long Beach CA*, 2009.

[15] Marcos de Melo da Silva, Güneş Erdoğan, Maria Battarra, and Vitaly Strusevich. The Block Retrieval Problem. *European Journal of Operational Research*, 265(3):931–950, 2018.

[16] Alberto Delgado, Rune Møller Jensen, Kira Janstrup, Trine Høyer Rose, and Kent Høj Andersen. A Constraint Programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research*, 220(1):251–261, 2012.

[17] Robert F. Dell, Johannes O. Royset, and Ioannis Zyngiridis. Optimizing container movements using one and two automated stacking cranes. *Journal of Industrial and Management Optimization*, 5(2):285–302, 2009.

[18] H Eskandari and E Azari. Notes on mathematical formulation and complexity considerations for blocks relocation problem. *Scientia Iranica. Transaction E, Industrial Engineering*, 22(6):2722–2728, 2015.

[19] C. Expósito-Izquierdo, B. Melián-Batista, and J. M. Moreno-Vega. Pre-Marshalling Problem: Heuristic solution method and instances generator. *Expert Systems with Applications*, 39(9):8337–8349, 2012.

[20] C. Expósito-Izquierdo, B. Melián-Batista, and J. M. Moreno-Vega. An exact approach for the Blocks Relocation Problem. *Expert Systems with Applications*, 42(17):6408–6422, 2015.

[21] F. Forster and A. Bortfeldt. A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39(2):299–309, 2012.

[22] V. Galle, C. Barnhart, and P. Jaillet. A New Binary Formulation of the Restricted Container Relocation Problem Based on a Binary Encoding of Congurations. *accepted for publication in European Journal of Operational Research on 11.27.2017*, 2017.

[23] V. Galle, C. Barnhart, and P. Jaillet. Yard crane scheduling for container storage, retrieval, and relocation. *working paper*, 2017.

[24] V. Galle, S. Borjian, V.H. Manshadi, C. Barnhart, and P. Jaillet. The Stochastic Container Relocation Problem. *submitted manuscript*, 2017. (Accessed 11.01.2017).

[25] V. Galle, S. Borjian Boroujeni, V.H. Manshadi, C. Barnhart, and P. Jaillet. An Average-Case Asymptotic Analysis of the Container Relocation Problem. *Operations Research Letters*, 44(6):723–728, 2016.

[26] Amir Hossein Gharehgozli, Gilbert Laporte, Yugang Yu, and René de Koster. Scheduling Twin Yard Cranes in a Container Block. *Transportation Science*, 49(3):686–705, 2015.

[27] Amir Hossein Gharehgozli, Debjit Roy, and René de Koster. Sea container terminals: New technologies and OR models. *Maritime Economics & Logistics*, 18(2):103–140, 2016.

[28] Amir Hossein Gharehgozli, Yugang Yu, René de Koster, and Jan Tijmen Udding. An exact method for scheduling a yard crane. *European Journal of Operational Research*, 235(2):431–447, 2014.

[29] Amir Hossein Gharehgozli, Yugang Yu, Xiandong Zhang, and René de Koster. Polynomial Time Algorithms to Minimize Total Travel Time in a Two-Depot Automated Storage/Retrieval System. *Transportation Science*, 51(1):19–33, 2017.

[30] Giovanni Giallombardo, Luigi Moccia, Matteo Salani, and Ilaria Vacca. Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B: Methodological*, 44(2):232–245, 2010.

[31] Genevieve Giuliano and Thomas O'Brien. Reducing port-related truck emissions: The terminal gate appointment system at the Ports of Los Angeles and Long Beach. *Transportation Research Part D: Transport and Environment*, 12(7):460–473, 2007.

[32] Xi Guo, Shell Ying Huang, Wen Jing Hsu, and Malcolm Yoke Hean Low. Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information. *Advanced Engineering Informatics*, 25(3):472–484, 2011.

[33] M. Hakan Akyüz and Chung-Yee Lee. A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Research Logistics (NRL)*, 61(2):101–118, 2014.

[34] Mark Huber. Exact Sampling from Perfect Matchings of Dense Regular Bipartite Graphs. *Algorithmica*, 44(3):183–193, 2006.

[35] Bo Jin, Wenbin Zhu, and Andrew Lim. Solving the container relocation problem by an improved greedy look-ahead heuristic. *European Journal of Operational Research*, 240(3):837 – 847, 2015.

[36] Michael Jünger, Thomas M Liebling, Denis Naddef, George L Nemhauser, William R Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A Wolsey. *50 years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009.

[37] Nils Kemme. Container-Terminal Logistics. In *Design and Operation of Automated Container Storage Systems*, pages 9–52. Physica-Verlag HD, 2013.

[38] Kap Hwan Kim and Gyu-Pyo Hong. A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954, 2006.

[39] Kap Hwan Kim and Ki Young Kim. An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science*, 33(1):17–33, 1999.

[40] Kap Hwan Kim, Young Man Park, and Kwang-Ryul Ryu. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124(1):89–101, 2000.

[41] Ki Young Kim and Kap Hwan Kim. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics (NRL)*, 50(5):498–514, 2003.

[42] Dusan Ku and Tiru S. Arthanari. On the abstraction method for the container relocation problem. *Computers & Operations Research*, 68(Supplement C):110–122, 2016.

[43] Dusan Ku and Tiru S. Arthanari. Container relocation problem with time windows for container departure. *European Journal of Operational Research*, 252(3):1031–1039, 2016.

[44] Der-Horng Lee, Zhi Cao, and Qiang Meng. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics*, 107(1):115–124, 2007.

[45] Yusin Lee and Shih-Liang Chao. A neighborhood search heuristic for pre-marshalling export containers. *European Journal of Operational Research*, 196(2):468–475, 2009.

[46] Yusin Lee and Nai-Yun Hsu. An optimization model for the container pre-marshalling problem. *Computers & Operations Research*, 34(11):3295–3313, 2007.

[47] Yusin Lee and Yen-Ju Lee. A heuristic for retrieving containers from a yard. *Computers & Operations Research*, 37(6):1139–1147, 2010.

[48] Pasquale Legato, Roberto Trunfio, and Frank Meisel. Modeling and Solving Rich Quay Crane Scheduling Problems. *Computers & Operations Research*, 39(9):2063–2078, 2012.

[49] Jana Lehnfeld and Sigrid Knust. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297–312, 2014.

[50] Israel López-Plata, Expósito-Izquierdo Christopher, Eduardo Lalla-Ruiz, Belén Melián-Batista, and J. Marcos Moreno-Vega. Minimizing the Waiting Times of block retrieval operations in stacking facilities. *Computers & Industrial Engineering*, 103(2):70–84, 2017.

[51] J. Matoušek and J. Vondrák. *The Probabilistic Method: Lecture Notes*. 2008. (Accessed 29.08.16).

[52] Philippe Morais and Elisabeth Lord. Terminal Appointment System Study. *Technical Report, Transportation Development Center of Transport Canada*, 2006. (Accessed 11.01.2017).

[53] Katta G Murty, Yat-wah Wan, Jiyin Liu, Mitchell M Tseng, Edmond Leung, Kam-Keung Lai, and Herman WC Chiu. Hongkong International Terminals gains elastic capacity using a data-intensive decision-support system. *Interfaces*, 35(1):61–75, 2005.

[54] Ananthapadmanabhan Narasimhan and Udatta S Palekar. Analysis and algorithms for the transtainer routing problem in container port operations. *Transportation Science*, 36(1):63–78, 2002.

[55] WC Ng and KL Mak. Yard crane scheduling in port container terminals. *Applied mathematical modelling*, 29(3):263–276, 2005.

[56] Martin Olsen and Allan Gross. Average Case Analysis of Blocks Relocation Heuristics. In *Computational Logistics: 5th International Conference, ICCL 2014, Valparaiso, Chile, September 24-26, 2014. Proceedings*, pages 81–92. 2014.

[57] Taejin Park, Ri Choe, Seung Min Ok, and Kwang Ryel Ryu. Real-time scheduling for twin RMGs in an automated container yard. *OR Spectrum*, 32(3):593–615, 2010.

[58] Matthew E.H. Petering and Mazen I. Hussein. A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *European Journal of Operational Research*, 231(1):120–130, 2013.

[59] E. E. Phillips. Southern California Ports to try Trucking Appointment System. *The Wall Street Journal*, 32(3):593–615, 2015. (Accessed 11.01.2017).

[60] Rui Rei and João Pedro Pedroso. Tree search for the stacking problem. *Annals of Operations Research*, 203(1):371–388, 2013.

[61] S. Saurí and E. Martín. Space allocating strategies for improving import yard performance at marine terminals. *Transportation Research Part E: Logistics and Transportation Review*, 47(6):1038–1057, 2011.

[62] Linn I Sennott. *Stochastic dynamic programming and the control of queueing systems*, volume 504. John Wiley & Sons, 2009.

[63] Ulf Speer and Kathrin Fischer. Scheduling of Different Automated Yard Crane Systems at Container Terminals. *Transportation Science*, 51(1):305–324, 2017.

[64] Robert Stahlbock and Stefan Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2008.

[65] Dirk Steenken, Stefan Voß, and Robert Stahlbock. Container terminal operation and operations research - a classification and literature review. *OR Spectrum*, 26(1):3–49, 2004.

[66] S Tanaka and K Takii. A Faster Branch-and-Bound Algorithm for the Block Relocation Problem. *IEEE Transactions on Automation Science and Engineering*, 13(1):181–190, 2016.

[67] Lixin Tang, Wei Jiang, Jiyin Liu, and Yun Dong. Research into container reshuffling and stacking problems in container terminal yards. *IIE Transactions*, 47(7):751–766, 2015.

[68] Lixin Tang, Ren Zhao, and Jiyin Liu. Models and algorithms for shuffling problems in steel plants. *Naval Research Logistics (NRL)*, 59(7):502–524, 2012.

[69] Kevin Tierney and Stefan Voß. Solving the Robust Container Pre-Marshalling Problem. In *Computational Logistics: 7th International Conference, ICCL 2016, Lisbon, Portugal, September 7-9, 2016, Proceedings*, pages 131–145. 2016.

[70] Fabien Tricoire, Judith Scagnetti, and Andreas Beham. New insights on the block relocation problem. *Computers & Operations Research*, 89(Supplement C):127–139, 2018.

[71] Tonguç Ünlüyurt and Cenk Aydın. Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation*, 46(4):378–393, 2012.

[72] Eelco van Asperen, Bram Borgman, and Rommert Dekker. Evaluating impact of truck announcements on container stacking efficiency. *Flexible Services and Manufacturing Journal*, 25(4):543–556, 2013.

[73] Iris FA Vis and Kees Jan Roodbergen. Scheduling of container storage and retrieval. *Operations Research*, 57(2):456–467, 2009.

[74] Yat-wah Wan, Jiyin Liu, and Pei-Chun Tsai. The assignment of storage locations to containers for a container stack. *Naval Research Logistics (NRL)*, 56(8):699–713, 2009.

[75] Jörg Wiese, Leena Suhl, and Natalia Kliewer. Mathematical models and solution methods for optimal container terminal yard layouts. *OR Spectrum*, 32(3):427–452, 2010.

[76] K.-C. Wu and C.-J. Ting. A beam search algorithm for minimizing reshuffle operations at container yards. In *Proceedings of the International Conference on Logistics and Maritime Systems*, pages 703–710, 2010.

[77] K.-C. Wu and C.-J. Ting. Heuristic approaches for minimizing reshuffle operations at container yard. In *Proceedings of the Asia Pacific industrial engineering & management systems conference*, pages 1407–1451, 2012.

[78] Dongsheng Xu, Chung-Lun Li, and Joseph Y.-T. Leung. Berth allocation with time-dependent physical limitations on vessels. *European Journal of Operational Research*, 216(1):47–56, 2012.

[79] Mingzhu Yu and Xiangtong Qi. Storage space allocation models for inbound containers in an automatic container terminal. *European Journal of Operational Research*, 226(1):32–45, 2013.

[80] Yuan Yuan and Lixin Tang. Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse. *European Journal of Operational Research*, 262(2):424–437, 2017.

[81] Elisabeth Zehendner, Marco Casserta, Dominique Feillet, Silvia Schwarze, and Stefan Voß. An improved mathematical formulation for the blocks relocation problem. *European Journal of Operational Research*, 245(2):415–422, 2015.

[82] Elisabeth Zehendner and Dominique Feillet. Benefits of a truck appointment system on the service quality of inland transport modes at a multimodal container terminal. *European Journal of Operational Research*, 235(2):461–469, 2014.

[83] Elisabeth Zehendner and Dominique Feillet. A branch and price approach for the container relocation problem. *International Journal of Production Research*, 52(24):7159–7176, 2014.

[84] Elisabeth Zehendner, Dominique Feillet, and Patrick Jaillet. An algorithm with performance guarantee for the Online Container Relocation Problem. *European Journal of Operational Research*, 259(1):48–62, 2017.

[85] Canrong Zhang, Weiwei Chen, Leyuan Shi, and Li Zheng. A note on deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 205(2):483–485, 2010.

[86] Wenjuan Zhao and Anne V. Goodchild. The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):327–343, 2010.

[87] Wenbin Zhu, Hu Qin, A. Lim, and Huidong Zhang. Iterative Deepening A* Algorithms for the Container Relocation Problem. *IEEE Transactions on Automation Science and Engineering*, 9(4):710–722, 2012.

Appendix A

Appendix on the Container Relocation Problem

A.1 Extensions of CRP-I

A.1.1 First Extension: Non-Uniform Relocations

Suppose that moving some containers has a higher cost than other containers, i.e., let R_d be the cost of relocating container d. Then CRP-I can be used to solve this problem just by considering the same constraints and the modified objective function:

$$Min \left(\sum_{n=1}^{N-1} \sum_{d=n+1}^{C} R_d \times a_{n,n,d} + \sum_{d=N+1}^{C} R_d \times b_d \right).$$

A.1.2 Second Extension: Minimizing Crane Travel Time

In this version of the CRP, the goal is to minimize the travel time of the crane (still in the restricted setting). This extension requires that we are given:

- the initial position of the crane. We suppose here that the crane starts at the position where it delivers containers to trucks (also called the I/O-point). This can easily be extended to any starting position.
- $t_{s,r}^{(1)}$ (respectively, $t_s^{(2)}$), the time to relocate a container from stack s to stack r

(respectively, the time to retrieve a container from stack s).

To extend CRP-I to this case (same as in [71]), we need to withdraw variable b and Constraint (3.8), take $N = C$ and introduce the binary variable:

$$\forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1, C\}, s \in \{1,\ldots,S\}, r \in \{1,\ldots,S\},$$
$$z_{n,c,s,r} = \begin{cases} 1 & \text{if container } c \text{ ``goes'' from stack } s \text{ to stack } r \text{ when } n \text{ is the target container,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that $z_{n,c,s,s} = 1$ means that container c is not relocated when container n is the target container, hence for the sake of clarity, we define $t^{(1)}_{s,s} = 0$. CRP-I can be adapted for minimizing crane travel time as follows:

$$Min \left(\sum_{n=1}^{N-1} \sum_{c=n+1}^{C} \sum_{s=1}^{S} \sum_{r=1}^{S} 2 \times t^{(1)}_{s,r} z_{n,c,s,r} + \sum_{n=1}^{N} \sum_{s=1}^{S} 2 \times t^{(2)}_{s} a_{n,C+s,n} \right)$$

s.t.

$(3.1) - (3.7), (3.9) - (3.13)$

$$\sum_{r=1}^{S} z_{n,c,s,r} = a_{n,C+s,c}, \qquad \forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1,\ldots,C\}, s \in \{1,\ldots,S\} \qquad \text{(A.1)}$$

$$\sum_{r=1}^{S} z_{n,c,r,s} = a_{n+1,C+s,c}, \qquad \forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1,\ldots,C\}, s \in \{1,\ldots,S\} \qquad \text{(A.2)}$$

$$z_{n,c,s,r} \in \{0,1\} \qquad \forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1, C\}, s \in \{1,\ldots,S\}, r \in \{1,\ldots,S\} \qquad \text{(A.3)}$$

The objective function is the total travel time of the crane, the first term accounts for relocations and the second term for retrievals. Moreover, we only have to add three types of constraints. Constraint (A.1) ensures that container c starts at stack s only if it is stack in s when n is the target container. Similarly, Constraint (A.2)

enforces that container c ends at stack s only if it is in stack s when $n+1$ is the target container. Finally, variable z is binary given Constraint (A.3).

A.1.3 Third Extension: the "Relaxed Restricted" CRP

Inspired from the unrestricted CRP which allows for relocations from a stack not containing the target container, also called cleaning moves (see [10, 58]), we propose here a simpler variant called the "relaxed restricted" CRP, where one cleaning move is allowed per container and per retrieval. In this version, CRP-I uses an extra type of variable defined as

$$\forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1, C\},$$
$$x_{n,c} = \begin{cases} 1 & \text{if container } c \text{ is relocated when } n \text{ is the target container,} \\ 0 & \text{otherwise.} \end{cases}$$

CRP-I can be adapted to the relaxed restricted case as follows:

$$Min \left(\sum_{n=1}^{N-1} \sum_{c=n+1}^{C} x_{n,c} + \sum_{d=N+1}^{C} b_d \right)$$

s.t.

$$(3.1) - (3.8), (3.13) - (3.14)$$

$$a_{n+1,d,c} \leqslant a_{n,d,c} + x_{n,c}, \quad \forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1,\ldots,C\}, d \in \{n+1,\ldots,C+S\} \setminus \{c\} \tag{3.9'}$$

$$a_{n+1,d,c} \geqslant a_{n,d,c} - x_{n,c}, \quad \forall\, n \in \{1,\ldots,N-1\}, c \in \{n+1,\ldots,C\}, d \in \{n+1,\ldots,C+S\} \setminus \{c\} \tag{3.10'}$$

$$x_{n,c} + a_{n,C+s,c} + a_{n+1,C+s,c} \leqslant 2, \quad \forall\, n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, s \in \{1,\dots,S\} \tag{3.11'}$$

$$x_{n,c} + x_{n,d} + a_{n,c,d} + a_{n+1,c,d} \leqslant 3, \quad \forall\, n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\}, d \in \{n+1,\dots,C\} \setminus \{c\}. \tag{3.12'}$$

$$x_{n,c} \geqslant a_{n,n,c}, \qquad \forall\, n \in \{1,\dots,N-1\}, c \in \{n+1,\dots,C\} \tag{A.4}$$

$$a_{n,c,d} + a_{n,e,f} + a_{n+1,e,d} + a_{n+1,c,f} \leqslant 3 + a_{n,e,d}, \quad \forall\, n \in \{1,\dots,N-1\}, c,e \in \{n+1,\dots,C+S\}, d,f \in \{n+1,\dots,C\}, c \neq d \neq e \neq f, \tag{A.5}$$

$$x_{n,c} \in \{0,1\} \qquad \forall\, n \in \{1,\dots,N-1\}, c \in \{n+1,C\} \tag{A.6}$$

First, $a_{n,n,c}$ has to be replaced by $x_{n,c}$. Moreover, we only have to add three types of constraints. Constraint (A.4) enforces that containers that are blocking the target container must be relocated. Constraint (A.5) ensures the LIFO policy holds for two different stacks. Consider two distinct stacks with at least one container in each (hence, two containers in each with the artificial container). Let c, d (respectively, e, f) be the containers in the first (respectively, second) stack with c (respectively, e) being below d (respectively, f). If these containers are indeed in two different stacks (i.e., $a_{e,d} = 0$), then Constraint (A.5) enforces that both c below f and e below d are not compatible after n has been retrieved. Finally, Constraint (A.6) ensures that variable x is binary. As a final remark, the relaxed restricted CRP is of interest for practice as it is very hard for port operators to perform several cleaning moves of one containers during busy periods. Even though, if theoretically in the unrestricted CRP, a container may be relocated more than once when n is the target container, we have noticed by comparing results with Petering and Hussein [58], on small instances

from [9] for $S = 4, 5$ and $T = 3, 4$, that the optimal solutions of the relaxed restricted CRP were also optimal for the unrestricted CRP. Future research questions include finding theoretical conditions under which the relaxed restricted optimal solution is also optimal for the unrestricted CRP. Another interesting question would be to adapt the binary encoding to the fully unrestricted setting.

A.2 Proof of Lemma 3

Lemma 3. *Let $h, S \in \mathbb{N}$ such that $S \geqslant h+1$ and $\Omega_{h,S}$ be the event defined by equation (3.18), then we have*

$$\mathbb{P}(\overline{\Omega_{h,S}}) \leqslant e^{-\theta_h(S+1)}, \tag{3.19}$$

where

$$\theta_h = \frac{1}{8h}\left(\frac{2}{h(h+1)}\right)^{2h} > 0. \tag{3.20}$$

Proof. Recall that

$$\Omega_{h,S} = \{B_{h,S+1} \text{ has at least one "special" stack}\}.$$

We know that each configuration of size $S+1$ can be mapped to a permutation π of $\mathcal{S}_{h(S+1)}$ taken uniformly at random. Let $q(.)$ be the function from $\mathcal{S}_{h(S+1)}$ to $\mathbb{R}^+$ defined by

$$q : \pi \longmapsto \text{number of "special" stacks in the resulting configuration of } \pi.$$

Note that

$$\mathbb{P}\left(\overline{\Omega_{h,S}}\right) = \mathbb{P}\left(q(\pi) = 0\right).$$

First we compute the expected value of $q(.)$

$$\begin{aligned}\mathbb{E}_{h,S+1}[q] = &\ \mathbb{E}_{h,S+1}\left[\sum_{i=1}^{S+1}\chi\left(s_i \text{ is a "special" stack}\right)\right] \\ = &\ (S+1)\times\mathbb{P}\left(\{s_1 \text{ is a "special" stack}\}\right),\end{aligned}$$

where we use linearity of expectation and the fact that stacks are identically distributed.

A simple counting implies that:

$$\begin{aligned}
&\mathbb{P}\left(\{s_1 \textit{ is a ``special'' stack}\}\right) \\
&= \frac{(S+1)[(S+1)-1]\dots[(S+1)-h+1]}{h(S+1)[h(S+1)-1]\dots[h(S+1)-h+1]} \\
&\geqslant \left(\frac{(S+1)-h+1}{h(S+1)}\right)^h \geqslant \left(\frac{2}{h(h+1)}\right)^h,
\end{aligned}$$

where we use $S+1 \geqslant h+1$ to show the last inequality (Notice that when $S \to \infty$, the probability is equivalent to $(1/h)^h$ which would guarantee a faster convergence rate).

Therefore we know that

$$\mathbb{E}_{h,S+1}[q] \geqslant (S+1) \times \left(\frac{2}{h(h+1)}\right)^h. \tag{A.7}$$

We claim that $q(.)$ is well concentrated around its mean. To do so, we prove that $q(.)$ is 1-Lipschitz.

Define ρ the distance between two permutations $\pi_1, \pi_2 \in \mathcal{S}_{h(S+1)}$ as

$$\rho(\pi_1, \pi_2) = |\left\{i \in [h(S+1)] : \pi_1(i) \neq \pi_2(i)\right\}|.$$

We want to prove that

$$|q(\pi_1) - q(\pi_2)| \leqslant \rho(\pi_1, \pi_2), \forall\, (\pi_1, \pi_2) \in \mathcal{S}_{h(S+1)}.$$

Let $\pi_1, \pi_2 \in \mathcal{S}_{h(S+1)}$. Let us first consider the case where $\rho(\pi_1, \pi_2) = 2$. (Notice that if $\rho(\pi_1, \pi_2) \neq 0$ then $\rho(\pi_1, \pi_2) \geqslant 2$). In that case, we have $i, j \in \{1, \dots, n\}$ such that $\pi_1(i) = \pi_2(j)$ and $\pi_1(j) = \pi_2(i)$. Let $B^{(1)}$ and $B^{(2)}$ be the configurations generated by π_1 and π_2. Having $\rho(\pi_1, \pi_2) = 2$ corresponds to the fact that if we swap 2 containers in $B^{(1)}$, we get $B^{(2)}$, we denote those containers $c = \pi_1(i)$, and $d = \pi_1(j)$. We have three cases:

- c and d are both in "special" stacks in $B^{(1)}$. In this case, swapping them will not change anything since both their new stacks in $B^{(2)}$ will also be "special," hence $|q(\pi_1) - q(\pi_2)| = 0$.

- c and d are both in stacks that are not "special" stacks in $B^{(1)}$. If $c, d \geqslant \omega_{h,S}$ or $c, d < \omega_{h,S}$ then we do not create any new special stack in $B^{(2)}$. Now suppose that $c \geqslant \omega_{h,S}$ and $d < \omega_{h,S}$, then the stack of c in $B^{(2)}$ might be a "special" stack, but the stack of d in $B^{(2)}$ cannot be "special". Therefore in that case, $|q(\pi_1) - q(\pi_2)| \leqslant 1$.

- c is in a "special" stack in $B^{(1)}$ but d is not. Now we know that $c \geqslant \omega_{h,S}$. If $d < \omega_{h,S}$ then the stack of d in $B^{(2)}$ cannot be "special" but the stack of c might be and in that case $|q(\pi_1) - q(\pi_2)| \leqslant 1$. If $d \geqslant \omega_{h,S}$, then the stack of d in $B^{(2)}$ is "special" and the stack of c in $B^{(2)}$ is not "special" which gives us $|q(\pi_1) - q(\pi_2)| = 0$. Note that the proof is identical if d is in a "special" stack in $B^{(1)}$ but c is not.

So far we have shown that

$$\textit{If } \rho(\pi_1, \pi_2) = 2, \textit{ then } |q(\pi_1) - q(\pi_2)| \leqslant 1. \tag{A.8}$$

Now we suppose that $\rho(\pi_1, \pi_2) = k$ where $2 \leqslant k \leqslant h(S+1)$. Note that we can construct a sequence of permutations $(\pi'_1, \pi'_2, \ldots, \pi'_k)$ such that $\pi'_1 = \pi_1$, $\pi'_k = \pi_2$, and $\rho(\pi'_i, \pi'_{i+1}) = 2$. Now using this fact and equation (A.8),

$$\begin{aligned}
|q(\pi_1) - q(\pi_2)| &= \left| \sum_{i=1}^{k-1} q(\pi'_i) - q(\pi'_{i+1}) \right| \\
&\leqslant \sum_{i=1}^{k-1} |q(\pi'_i) - q(\pi'_{i+1})| \leqslant \sum_{i=1}^{k-1} 1 = k - 1 \\
&\leqslant k = \rho(\pi_1, \pi_2),
\end{aligned}$$

which proves that $q(.)$ is 1-Lipschitz.

Now we use Theorem 8.3.3 from [51] which states that

$$\mathbb{P}\left(q \leqslant \mathbb{E}_{h,S+1}[q] - t\right) \leqslant e^{-\dfrac{t^2}{8h(S+1)}},$$

and apply it with $t = \mathbb{E}_{h,S+1}[q]$ and equation (A.7) to get

$$\begin{aligned}
\mathbb{P}\left(q = 0\right) = &\ \mathbb{P}\left(q \leqslant \mathbb{E}_{h,S+1}[q] - \mathbb{E}_{h,S+1}[q]\right) \\
\leqslant &\ e^{-\dfrac{\left(\mathbb{E}_{h,S+1}[q]\right)^2}{8h(S+1)}} \\
\leqslant &\ e^{-\theta_h(S+1)},
\end{aligned}$$

where

$$\theta_h = \frac{1}{8h}\left(\frac{2}{h(h+1)}\right)^{2h} > 0,$$

which concludes the proof of Lemma 3. □

Appendix B

Appendix on the Stochastic Container Relocation Problem

B.1 Theoretical and Computational Comparison of the Batch and the Online Models

B.1.1 Theoretical Comparison: Proof of Lemma 4

Lemma 4. *Let y be a given initial configuration, then we have*

$$f(y) \leqslant f^o(y).$$

Proof. We prove this lemma by induction on the number of batches W. The lemma clearly holds if y is empty (i.e., $W = 0$). Now consider $W \geqslant 1$ and $C_1 \geqslant 1$. For the sake of clarity of the proof, we define the following notation:

$$\forall\, d \in \{1, \ldots, C_1\}\,, \begin{cases} y \xrightarrow{\zeta_1, \ldots, \zeta_d} x_1^d \\ x_k^d \xrightarrow{a_k} x_{k+1}^d, \text{ if } d > 1\ ,\ \forall\, k \in \{1, \ldots, d-1\} \\ x_k^d \xrightarrow{a_k} y_{k-d+2}^d\ ,\ \forall\, k \in \{d, \ldots, C\} \\ y_{k-d+1}^d \xrightarrow{\zeta_k} x_k^d\ ,\ \forall\, k \in \{d+1, \ldots, C\}. \end{cases} \tag{B.1}$$

These notations corresponds to the following process: the first d containers to be

retrieved are all revealed at once. Then decisions to retrieve these d containers are made. Afterwards, each of the $C-d$ remaining containers is revealed one at a time (as in the online model). Under this revelation process, the minimum expected number of relocation is given by

$$f^d(y) = \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_d}\left[\min_{a_1,\ldots,a_d}\left\{\sum_{k=1}^{d} r\left(x_k^d\right) + f^o\left(y_2^d\right)\right\}\right],\ \forall\ d \in \{1,\ldots,C_1\}.$$

Moreover, using the recursion formula from the online model, we have

$$f^o\left(y_2^d\right) = \mathop{\mathbb{E}}_{\zeta_d}\left[\min_{a_d}\left\{r\left(x_{d+1}^d\right) + f^o\left(y_3^d\right)\right\}\right].$$

In particular, by definition of the online model, we have $f^o(y) = f^1(y)$.

Using these relations, let us prove that

$$f^d(y) \leqslant f^{d-1}(y),\ \forall\ d \in \{2,\ldots,C_1\}. \tag{B.2}$$

Let $d \in \{2,\ldots,C_1\}$, we have

$$\begin{aligned}
f^d(y) &= \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_d}\left[\min_{a_1,\ldots,a_d}\left\{\sum_{k=1}^{d} r\left(x_k^d\right) + f^o\left(y_2^d\right)\right\}\right] \\
&= \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{d-1}}\left[\mathop{\mathbb{E}}_{\zeta_d}\left[\min_{a_1,\ldots,a_{d-1}}\left\{\min_{a_d}\left\{\sum_{k=1}^{d} r\left(x_k^d\right) + f^o\left(y_2^d\right)\right\}\right\}\right]\right] &\text{(B.3)}\\
&\leqslant \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{d-1}}\left[\min_{a_1,\ldots,a_{d-1}}\left\{\mathop{\mathbb{E}}_{\zeta_d}\left[\min_{a_d}\left\{\sum_{k=1}^{d} r\left(x_k^{d-1}\right) + f^o\left(y_3^{d-1}\right)\right\}\right]\right\}\right] &\text{(B.4)}\\
&= \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{d-1}}\left[\min_{a_1,\ldots,a_{d-1}}\left\{\sum_{k=1}^{d-1} r\left(x_k^{d-1}\right) + \mathop{\mathbb{E}}_{\zeta_d}\left[\min_{a_d}\left\{r\left(x_d^{d-1}\right) + f^o\left(y_3^{d-1}\right)\right\}\right]\right\}\right] &\text{(B.5)}\\
&= \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{d-1}}\left[\min_{a_1,\ldots,a_{d-1}}\left\{\sum_{k=1}^{d-1} r\left(x_k^{d-1}\right) + f^o\left(y_2^{d-1}\right)\right\}\right] = f^{d-1}(y),
\end{aligned}$$

where the equality (B.5) holds since x_k^{d-1} for $k \in \{1,\ldots,d-1\}$ does not depend on a_d and ζ_d. Finally, the inequality holds because we have $\mathbb{E}[\min\{Z_1,\ldots,Z_m\}] \leqslant \min\{\mathbb{E}[Z_1,\ldots,Z_m]\}$ for any $Z_1,\ldots,Z_m$ random variables. Note that we changed x_k^d

in x_k^{d-1} and y_2^d in y_3^{d-1}. This change is necessary to stay consistent with the definition of Equation (B.1). Indeed, the order between the expectations and the minimums in Equation (B.3) implies that the process of the first d retrievals corresponds to

$$y \xrightarrow{\zeta_1,\ldots,\zeta_d} x_1^d \xrightarrow{a_1} x_2^d \xrightarrow{a_2} \ldots \xrightarrow{a_{d-1}} x_d^d \xrightarrow{a_d} y_2^d,$$

whereas the order between the expectations and the minimums in Equation (B.4) corresponds to the following process for the first d retrievals:

$$y \xrightarrow{\zeta_1,\ldots,\zeta_{d-1}} x_1^{d-1} \xrightarrow{a_1} x_2^{d-1} \xrightarrow{a_2} \ldots \xrightarrow{a_{d-1}} y_2^{d-1} \xrightarrow{\zeta_d} x_d^{d-1} \xrightarrow{a_d} y_3^{d-1}.$$

Recall Equation (4.1) and apply it with $w = 1$ (note that $K_1 = 1$, thus $K_1 + C_1 - 1 = C_1$) to get

$$f(y) = \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{C_1}} \left[\min_{a_1,\ldots,a_{C_1}} \left\{ \sum_{k=1}^{C_1} r(x_k) + f(y_2) \right\} \right].$$

By induction, for all configuration y_2 with $W - 1$ batches we have $f(y_2) \leqslant f^o(y_2)$, thus

$$\begin{aligned} f(y) &= \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{C_1}} \left[\min_{a_1,\ldots,a_{C_1}} \left\{ \sum_{k=1}^{C_1} r(x_k) + f(y_2) \right\} \right] \\ &\leqslant \mathop{\mathbb{E}}_{\zeta_1,\ldots,\zeta_{C_1}} \left[\min_{a_1,\ldots,a_{C_1}} \left\{ \sum_{k=1}^{C_1} r\left(x_k^{C_1}\right) + f^o\left(y_2^{C_1}\right) \right\} \right] \\ &= f^{C_1}(y), \end{aligned}$$

where we replaced x_k by $x_k^{C_1}$ and y_2 by $y_2^{C_1}$ because, on the right-hand side of the inequality, the revelation process after the first C_1 containers is the online model. Finally, since $f^o(y) = f^1(y)$, by applying Equation (B.2) for each value of $d \in \{C_1, \ldots, 2\}$, we complete the proof as

$$f(y) \leqslant f^{C_1}(y) \leqslant f^{C_1-1}(y) \leqslant \ldots \leqslant f^2(y) \leqslant f^1(y) = f^o(y).$$

□

As a final remark, Lemma 4 is tight in the general setting. Indeed, there exists an initial configuration y for which $f(y) = f^o(y)$. For instance consider the configuration in Figure 4-2a, then we have $f(y) = f^o(y) = 13/6$.

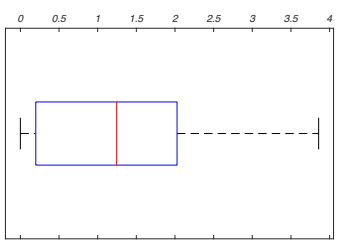
B.1.2 Computational Comparison

There also exist configurations for which $f(y) < f^o(y)$. The difference between these two values represents the value of taking into account available information (if possible).

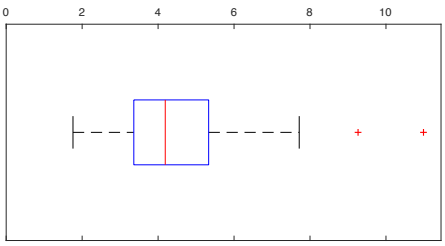
To show a positive difference, we could have compared Experiments 1 and 3. However, since the average batch size is two, these experiments do not show a positive difference between both models. Another possibility would have been to use the instances from Experiment 2. However, as we previously mentioned, such instances are hard to solve optimally and not approximately.

Instead, we consider another set of simpler instances randomly generated: 100 instances with $T = 4$ tiers, $S = 4$ stacks, and $C = 12$ containers. Each instance has $W = 3$ batches and each batch has $C_w = 4$ containers (for $w = 1, 2, 3$). We solve each of these 100 instances under the batch and the online models. The code and detailed results are available at https://github.com/vgalle/StochasticCRP. We are especially interested about $\frac{f^o(.) - f(.)}{f(.)} \times 100$, which the percentage difference between the batch and the online models.

On average over the 100 instances, the optimal expected number of relocations under the batch model is 6.526 and under the online model is 6.61, hence giving a difference of 0.084. We observe here that this difference represents more than 1.287 percent of the optimal solution under the batch model, which is quite significant considering the fact that heuristic EM experimentally lies within 2 percent above the optimal solution. In addition, we noticed that for 25 of these instances, this difference was more than 2 percent and the maximum of about 4 percent (see Figure B-1a).



B-1a $T = 4$, $S = 4$, and $C = 12$ with $W = 3$ and $C_w = 4$ (for $w = 1, 2, 3$).



B-1b $T = 4$, $S = 4$, and $C = 12$ with $W = 2$ and $C_w = 6$ (for $w = 1, 2$).

Figure B-1: Distributions of percentage difference between the batch and the online models from 100 randomly generated instances.

We also consider 100 instances for which $T = 4$, $S = 4$, and $C = 12$, but now $W = 2$ and each batch has $C_w = 6$ (for $w = 1, 2$). Figure B-1b shows that this relative difference appears to increase when the batch size increases. Indeed, the average difference is about 4.251 percent (batch: 6.751, online: 7.038, difference: 0.287), with 25 instances having a difference of more than 5.3 percent.

B.2 Proof of Lemma 8

Lemma 8. *Let n be a configuration with $\lambda_n \geqslant 0$ containers, l be a valid lower bound function, and $\epsilon > 0$. If $\tilde{f}(n) = PBFSA(n, l, \epsilon)$, then*

$$\mathbb{E}\left[\left|\tilde{f}(n) - f(n)\right|\right] \leqslant \epsilon.$$

Proof. The proof is by induction on λ_n. Throughout the proof, we use the same notations as the ones introduced in Algorithm 3. We say that $\tilde{f}$ verifies Conditions (A) and (B) at node n, if it verifies respectively the first and second inequalities below:

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+\right] \leqslant \frac{\epsilon}{2} \text{ and } \mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^-\right] \leqslant \frac{\epsilon}{2}.$$

Note that if $\tilde{f}$ verifies Conditions (A) and (B) at node n, then $\mathbb{E}\left[\left|\tilde{f}(n) - f(n)\right|\right] \leqslant \epsilon$, which would prove the lemma. Given $\epsilon > 0$, and l a valid lower bound, the induction

hypothesis is:

If $\tilde{f}(n) = PBFSA(n, l, \epsilon)$, then $\tilde{f}$ verifies Conditions (A) and (B) at node n.

First, if $\lambda_n \leqslant S$, then $\tilde{f}(n) = b(n) = f(n)$ and therefore, $\tilde{f}$ verifies Conditions (A) and (B) at node n. In this case, $\tilde{f}(n)$ is actually deterministic since no sampling is performed by *PBFSA*. From now on, consider n such that $\lambda_n > S$.

Analysis of error at decision nodes

If n is a decision node such that $\tilde{f}(n) = PBFSA(n, l, \epsilon)$ and $\lambda_n > S$. First, if $S < \lambda_n \leqslant C_W$, then $\tilde{f}(n) = A^*(n) = f(n)$, hence $\tilde{f}$ verifies Conditions (A) and (B) at n.

If $\lambda_n > \max\{S, C_W\}$, consider $\tilde{n} = \underset{n_i \in \Gamma_n^{PBFSA}}{argmin} \left\{\tilde{f}(n_i)\right\}$ and $n^* = \underset{n_i \in \Delta_n}{argmin} \{f(n_i)\}$. Note that $\tilde{f}(n) - f(n) = \tilde{f}(\tilde{n}) - f(n^*)$ almost surely $(a.s.)$. By definition $\tilde{n}$ and n^* are both such that $\lambda_{\tilde{n}} = \lambda_{n^*} = \lambda_n - 1 < \lambda_n$. Consider the following measurable event:

$$\mathcal{E} = \left\{\tilde{f}(n) - f(n) = \tilde{f}(\tilde{n}) - f(n^*) > 0\right\}. \tag{B.6}$$

- Conditioned on $\mathcal{E}$, we have $\left(\tilde{f}(n) - f(n)\right)^- = 0$ *a.s.*, thus

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^- \;\middle|\; \mathcal{E}\right] = 0. \tag{B.7}$$

  Now let us show that conditioned on $\mathcal{E}$, $n^* \in \Gamma_n^{PBFSA}$ *a.s.*; we suppose by contradiction that *a.s.* $n^* \notin \Gamma_n^{PBFSA}$. If $k = \left|\Gamma_n^{PBFSA}\right|$, then $k < |\Delta_n|$ *a.s.*, and $\min\limits_{j=1,\ldots,k} \left\{\tilde{f}(n_{(j)})\right\} \leqslant l\left(n_{(k+1)}\right)$ *a.s.* By definition $\tilde{f}(\tilde{n}) = \min\limits_{j=1,\ldots,k} \left\{\tilde{f}(n_{(j)})\right\}$ so $\tilde{f}(\tilde{n}) \leqslant l\left(n_{(k+1)}\right)$ *a.s.* Since

$n^* \notin \Gamma_n^{PBFSA}$, there exists $k^* \in \{k+1, \ldots, |\Delta_n|\}$ such that $n^* = n_{(k^*)}$. Since $\left(n_{(i)}\right)_{i \in \{1,\ldots,|\Delta_n|\}}$ are ordered by nondecreasing $l(.)$, we have $l\left(n_{(k+1)}\right) \leqslant l\left(n_{(k^*)}\right) = l\left(n^*\right)$. Therefore $\tilde{f}\left(\tilde{n}\right) \leqslant l\left(n^*\right)$ *a.s.*; but, conditioned on $\mathcal{E}$, $\tilde{f}\left(\tilde{n}\right) > f\left(n^*\right) \geqslant l\left(n^*\right)$ *a.s.*, which leads to a contradiction. Thus conditioned on $\mathcal{E}$, $n^* \in \Gamma_n^{PBFSA}$ *a.s.*. Therefore, we have $\tilde{f}(n^*) = PBFSA(n^*, l, \epsilon)$. By induction, $\tilde{f}$ verifies Condition (A) at node n^*, thus

$$\mathbb{E}\left[\left(\tilde{f}(n^*) - f(n^*)\right)^+\right] \leqslant \frac{\epsilon}{2}. \tag{B.8}$$

Finally, since $\tilde{n} = \underset{n_i \in \Gamma_n^{PBFSA}}{argmin} \left\{\tilde{f}(n_i)\right\}$ and $n^* \in \Gamma_n^{PBFSA}$, then $\tilde{f}(\tilde{n}) \leqslant \tilde{f}(n^*)$ *a.s.*, so we have $\tilde{f}(\tilde{n}) - f(n^*) \leqslant \tilde{f}(n^*) - f(n^*)$ *a.s.* Consequently we have $\left(\tilde{f}(n) - f(n)\right)^+ = \left(\tilde{f}(\tilde{n}) - f(n^*)\right)^+ \leqslant \left(\tilde{f}(n^*) - f(n^*)\right)^+$ *a.s.*, resulting in

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+ \;\middle|\; \mathcal{E}\right] \leqslant \mathbb{E}\left[\left(\tilde{f}(n^*) - f(n^*)\right)^+ \;\middle|\; \mathcal{E}\right]. \tag{B.9}$$

- Conditioned on $\overline{\mathcal{E}}$, we have $\left(\tilde{f}(n) - f(n)\right)^+ = 0$ *a.s.*, thus

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+ \;\middle|\; \overline{\mathcal{E}}\right] = 0. \tag{B.10}$$

Moreover, by definition $\tilde{n} \in \Gamma_n^{PBFSA}$ *a.s.*, and $\tilde{f}(\tilde{n}) = PBFSA(\tilde{n}, l, \epsilon)$, thus the induction hypothesis can be applied to $\tilde{n}$. In particular, we have

$$\mathbb{E}\left[\left(\tilde{f}(\tilde{n}) - f(\tilde{n})\right)^-\right] \leqslant \frac{\epsilon}{2}. \tag{B.11}$$

Finally, it is clear that $f(\tilde{n}) \geqslant f(n^*)$, then $\tilde{f}(\tilde{n}) - f(n^*) \geqslant \tilde{f}(\tilde{n}) - f(\tilde{n})$ *a.s.*, which is equivalent to $\left(\tilde{f}(n) - f(n)\right)^- = \left(\tilde{f}(\tilde{n}) - f(n^*)\right)^- \leqslant \left(\tilde{f}(\tilde{n}) - f(\tilde{n})\right)^-$ *a.s.*, resulting in

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^- \;\middle|\; \overline{\mathcal{E}}\right] \leqslant \mathbb{E}\left[\left(\tilde{f}(\tilde{n}) - f(\tilde{n})\right)^- \;\middle|\; \overline{\mathcal{E}}\right]. \quad \text{(B.12)}$$

Finally, note the following observation: Let $Y \geqslant 0$ *a.s.*, and $\mathcal{F}$ be measurable, then we have

$$\mathbb{E}\left[Y \mid \mathcal{F}\right]\mathbb{P}\left(\mathcal{F}\right) \leqslant \mathbb{E}\left[Y\right] \text{ and } \mathbb{E}\left[Y \mid \overline{\mathcal{F}}\right]\mathbb{P}\left(\overline{\mathcal{F}}\right) \leqslant \mathbb{E}\left[Y\right].$$

Now we can derive

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+\right] &= \mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+ \;\middle|\; \mathcal{E}\right]\mathbb{P}\left(\mathcal{E}\right) \\
&\leqslant \mathbb{E}\left[\left(\tilde{f}(n^*) - f(n^*)\right)^+ \;\middle|\; \mathcal{E}\right]\mathbb{P}\left(\mathcal{E}\right) \\
&\leqslant \mathbb{E}\left[\left(\tilde{f}(n^*) - f(n^*)\right)^+\right] \\
&\leqslant \frac{\epsilon}{2},
\end{aligned}$$

where the first equality comes from Equation (B.10), the first inequality uses Equation (B.9), the second one holds thanks to $\left(\tilde{f}(n^*) - f(n^*)\right)^+ \geqslant 0$ *a.s.*, and the last one is Equation (B.8). Therefore, $\tilde{f}$ verifies Condition (A) at node n.

Similarly, we have

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{f}(n)-f(n)\right)^{-}\right] &= \mathbb{E}\left[\left(\tilde{f}(n)-f(n)\right)^{-} \;\middle|\; \overline{\mathcal{E}}\right]\mathbb{P}\left(\overline{\mathcal{E}}\right) \\
&\leqslant \mathbb{E}\left[\left(\tilde{f}(\tilde{n})-f(\tilde{n})\right)^{-} \;\middle|\; \overline{\mathcal{E}}\right]\mathbb{P}\left(\overline{\mathcal{E}}\right) \\
&\leqslant \mathbb{E}\left[\left(\tilde{f}(\tilde{n})-f(\tilde{n})\right)^{-}\right] \\
&\leqslant \frac{\epsilon}{2},
\end{aligned}$$

where the first equality comes from Equation (B.7), the first inequality uses Equation (B.12), the second one holds thanks to $\left(\tilde{f}(\tilde{n})-f(\tilde{n})\right)^{-} \geqslant 0$ *a.s.*, and the last one is Equation (B.11). Therefore, $\tilde{f}$ verifies Condition (B) at node n.

Therefore, we have proven that if n is a decision node with $\lambda_n > S$, $\tilde{f}$ verifies both Conditions (A) and (B) at node n, which proves the lemma for decision nodes.

Analysis of error at chance nodes

If n is a chance node such that $\tilde{f}(n) = PBFSA(n, l, \epsilon)$, and $\lambda_n > S$. Let us define $\overline{f}(n) = \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n f(n_i)$, and show that

$$\mathbb{E}\left[\left(\tilde{f}(n)-\overline{f}(n)\right)^{+}\right] \leqslant \frac{\epsilon-\epsilon_n}{2} \text{ and } \mathbb{E}\left[\left(\tilde{f}(n)-\overline{f}(n)\right)^{-}\right] \leqslant \frac{\epsilon-\epsilon_n}{2}. \tag{B.13}$$

Recall that $\forall n_i \in \Psi_n^{PBFSA}$, $\lambda_{n_i} = \lambda_n$, and n_i are decision nodes such that $\tilde{f}(n_i) = PBFSA(n_i, l, \epsilon - \epsilon_n)$. Therefore, using the previous result, we know that $\mathbb{E}\left[\left(\tilde{f}(n_i)-f(n_i)\right)^{+}\right] \leqslant \frac{\epsilon-\epsilon_n}{2}$ and $\mathbb{E}\left[\left(\tilde{f}(n_i)-f(n_i)\right)^{-}\right] \leqslant$

$\frac{\epsilon - \epsilon_n}{2}$. We derive the following calculations:

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{f}(n) - \overline{f}(n)\right)^+\right] &= \mathbb{E}\left[\left(\sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \left(\tilde{f}(n_i) - f(n_i)\right)\right)^+\right] \\
&\leqslant \mathbb{E}\left[\sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \left(\tilde{f}(n_i) - f(n_i)\right)^+\right] \\
&= \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \mathbb{E}\left[\left(\tilde{f}(n_i) - f(n_i)\right)^+\right] \\
&\leqslant \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \frac{\epsilon - \epsilon_n}{2} = \frac{\epsilon - \epsilon_n}{2}.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{f}(n) - \overline{f}(n)\right)^-\right] &= \mathbb{E}\left[\left(\sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \left(\tilde{f}(n_i) - f(n_i)\right)\right)^-\right] \\
&\leqslant \mathbb{E}\left[\sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \left(\tilde{f}(n_i) - f(n_i)\right)^-\right] \\
&= \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \mathbb{E}\left[\left(\tilde{f}(n_i) - f(n_i)\right)^-\right] \\
&\leqslant \sum_{n_i \in \Psi_n^{PBFSA}} p_{n_i}^n \frac{\epsilon - \epsilon_n}{2} = \frac{\epsilon - \epsilon_n}{2},
\end{aligned}$$

which proves Equation (B.13).

If $N_n(\epsilon_n) > C_{w_{min}}!$, then $f(n) = \overline{f}(n)$ so $\tilde{f}(n) - f(n) = \tilde{f}(n) - \overline{f}(n)$ *a.s.*, and since $\frac{\epsilon - \epsilon_n}{2} \leqslant \frac{\epsilon}{2}$, Equation (B.13) implies that $\tilde{f}$ verifies Conditions (A) and (B) at node n.

Otherwise, we have $N_n(\epsilon_n) \leqslant C_{w_{min}}!$. Since Ψ_n^{PBFSA} is constructed

using $N_n(\epsilon_n) = \dfrac{\pi \left(f_{max}(n) - f_{min}(n)\right)^2}{2\epsilon_n^2}$ samples, thus by using Corollary 2, we have

$$\mathbb{E}\left[\left(\overline{f}(n) - f(n)\right)^+\right] \leqslant \frac{\epsilon_n}{2} \text{ and } \mathbb{E}\left[\left(\overline{f}(n) - f(n)\right)^-\right] \leqslant \frac{\epsilon_n}{2} \tag{B.14}$$

By combining Equations (B.13) and (B.14), we have

$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^+\right] \leqslant \mathbb{E}\left[\left(\tilde{f}(n) - \overline{f}(n)\right)^+\right] + \mathbb{E}\left[\left(\overline{f}(n) - f(n)\right)^+\right] \leqslant \frac{\epsilon - \epsilon_n}{2} + \frac{\epsilon_n}{2} = \frac{\epsilon}{2},$$
$$\mathbb{E}\left[\left(\tilde{f}(n) - f(n)\right)^-\right] \leqslant \mathbb{E}\left[\left(\tilde{f}(n) - \overline{f}(n)\right)^-\right] + \mathbb{E}\left[\left(\overline{f}(n) - f(n)\right)^-\right] \leqslant \frac{\epsilon - \epsilon_n}{2} + \frac{\epsilon_n}{2} = \frac{\epsilon}{2},$$

which shows that $\tilde{f}$ verifies Conditions (A) and (B) at node n and concludes the proof. □

B.3 Technical Proofs of Section 4.6.1

Corollary 2. *Let $X \in [x_{min}, x_{max}]$ be a real-valued bounded random variable with mean value $\mathbb{E}[X]$. Let $N \in \mathbb{N}$ and $(X_1, \ldots, X_N)$ be N i.i.d. samples of X. If $\overline{X} = \dfrac{1}{N}\sum_{i=1}^{N} X_i$, then $\forall \epsilon > 0$ such that $N \geqslant \dfrac{\pi \left(x_{max} - x_{min}\right)^2}{2\epsilon^2}$, we have*

$$\mathbb{E}\left[\left(\overline{X} - \mathbb{E}[X]\right)^+\right] \leqslant \frac{\epsilon}{2}, \tag{4.7}$$

$$\mathbb{E}\left[\left(\overline{X} - \mathbb{E}[X]\right)^-\right] \leqslant \frac{\epsilon}{2}, \tag{4.8}$$

where $x^+ = \max\{x, 0\}$ (resp. $x^- = -\min\{x, 0\}$) is the positive (resp. negative) part of x.

Proof of Corollary 2. For the first result, define

$$\Delta = \left(\overline{X} - \mathbb{E}[X]\right)^+ = \left(\overline{X} - \mathbb{E}[X]\right) \mathbb{1}\left\{\overline{X} - \mathbb{E}[X] > 0\right\}.$$

Note that Δ is a non-negative random variable, and $\forall \delta > 0$,

$$\{\Delta > \delta\} = \left\{\overline{X} - \mathbb{E}[X] > \delta\right\}.$$

Let F_Δ denote the cumulative distribution function of Δ, thus, using Equation (4.5), $1 - F_\Delta(\delta) = \mathbb{P}(\Delta > \delta) = \mathbb{P}\left(\overline{X} - \mathbb{E}[X] > \delta\right) \leqslant exp\left(\frac{-2N\delta^2}{(x_{max}-x_{min})^2}\right)$, which gives

$$\begin{aligned}\mathbb{E}[\Delta] = & \int_{\delta=0}^{\infty} (1 - F_\Delta(\delta))\, d\delta \leqslant \int_{\delta=0}^{\infty} exp\left(\frac{-2N\delta^2}{(x_{max} - x_{min})^2}\right) d\delta \\ & = \frac{(x_{max} - x_{min})}{\sqrt{2N}} \int_{u=0}^{\infty} exp(-u^2) du = \frac{\sqrt{\pi}\,(x_{max} - x_{min})}{2\sqrt{2N}} \leqslant \frac{\epsilon}{2}.\end{aligned}$$

The proof of the second result is identical to the first one if we consider $\Delta' = \left(\overline{X} - \mathbb{E}[X]\right)^- = \left(\mathbb{E}[X] - \overline{X}\right) \mathbb{1}\left\{\mathbb{E}[X] - \overline{X} > 0\right\}$ and notice that $\forall \delta > 0$, $\{\Delta' > \delta\} = \left\{\overline{X} - \mathbb{E}[X] < -\delta\right\}$, hence $1 - F_{\Delta'}(\delta) \leqslant exp\left(\frac{-2N\delta^2}{(x_{max}-x_{min})^2}\right)$ using Equation (4.6). □

Lemma 9. *Let n be a chance node, if*

$$f_{min}(n) = \min_{n_i \in \Omega_n} \{b(n_i)\}, \tag{4.9}$$

and

$$f_{max}(n) = \min\left\{\left((\lambda_n - S)(T-1)\right)^+ + \left(\min\{S, \lambda_n\} - 1\right),\ \left(2\left\lceil\frac{\lambda_n}{S}\right\rceil - 1\right) \max_{n_i \in \Omega_n} \{b(n_i)\}\right\}, \tag{4.10}$$

then

$$f_{min}(n) \leqslant \min_{n_i \in \Omega_n} \{f(n_i)\} \text{ and } f_{max}(n) \geqslant \max_{n_i \in \Omega_n} \{f(n_i)\}.$$

Proof of Lemma 9. Since $b(n_i) \leqslant f(n_i)$, then we have $f_{min}(n) = \min_{n_i \in \Omega_n} \{b(n_i)\} \leqslant \min_{n_i \in \Omega_n} \{f(n_i)\}$.

By definition $f_{max}(n)$ is the minimum of two valid upper bounds. The first one comes from a basic observation. If there are λ_n containers remaining to be retrieved in n, consider two cases:

- If $\lambda_n > S$, take the rth retrieval. If $S < r \leqslant \lambda_n$, then to perform this retrieval, there are at most $T-1$ containers blocking the target container so at most $T-1$

relocations are needed. When there are S or fewer containers remaining, each container (except the lowest one) is at most relocated once, hence we need at most $S-1$ relocations. Combining these two facts, the maximum number of relocations is bounded by $(\lambda_n - S)(T-1) + (S-1) = ((\lambda_n - S)(T-1))^+ + (\min\{S, \lambda_n\} - 1)$.

- If $\lambda_n \leqslant S$, we know that $f(n_i) = b(n_i) \leqslant \lambda_n - 1 = ((\lambda_n - S)(T-1))^+ + (\min\{S, \lambda_n\} - 1)$.

This shows the validity of the first upper bound.

For the second upper bound, Zehendner et al. [84] prove that, in the online case with a unique batch, the number of relocations performed by the leveling heuristic (L) is at most $\left(2\left\lceil\frac{\lambda_n}{S}\right\rceil - 1\right)B$, where B is the number of blocking containers. Since L is not using any information about batches (only the height of stacks), this result holds for both batch and online models with any number of batches. Let $n_i \in \Omega_n$, using this result and taking expectation over the retrieval order of containers not unveiled in n_i yet, we have $f(n_i) \leqslant f_L(n_i) \leqslant \left(2\left\lceil\frac{\lambda_n}{S}\right\rceil - 1\right) b(n_i)$. By taking the maximum over all $n_i \in \Omega_n$, the latter inequality results in the second upper bound. □

Lemma 10. *Let n be a chance node, and $w_{min} \in \{1, \dots, W\}$ be such that $\lambda_n = C - K_{w_{min}} + 1$ (i.e., the minimum batch in n). For each Stack s of n with $H^s \geqslant 1$ containers, let $(c^s_h)_{h=1,\dots,H^s}$ be the containers in s, where c^s_1 is the container at the bottom and $c^s_{H^s}$ at the top (see Figure 4-9, for the case $H = H^s$). Finally, consider $C^s_{w_{min}} = |\{c^s_h = K_{w_{min}},\ h = 1, \dots, H^s\}|$. Then we have*

$$\min_{n_i \in \Omega_n}\{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - C^s_{w_{min}} - \sum_{\substack{h=1,\dots,H^s \\ c^s_h \neq K_{w_{min}}}} \frac{\mathbb{1}\left\{c^s_h = \min\limits_{i=1,\dots,h}\{c^s_i\}\right\}}{\sum\limits_{i=1}^{h} \mathbb{1}\{c^s_h = c^s_i\}} \right), \quad (4.11)$$

and

$$\max_{n_i \in \Omega_n} \{b(n_i)\} = \sum_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} \left(H^s - \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} \frac{\mathbb{1}\left\{c_h^s = \min\limits_{i=1,\dots,h} \{c_i^s\}\right\}}{\sum\limits_{i=1}^{h} \mathbb{1}\{c_h^s = c_i^s\}} \right). \tag{4.12}$$

Proof of Lemma 10. Let n be a chance node, and $n_i \in \Omega_n$ be one of its decision offspring, such that all containers in batch w_{min} have been revealed. Recall that $b(n_i) = \sum\limits_{s=1,\dots,S} b^s(n_i)$, where $b^s(n_i)$ is the expected number of blocking containers in Stack s. First, for each Stack s such that $H^s = 0$, $b^s(n_i) = 0$. Hence $b(n_i) = \sum\limits_{\substack{s=1,\dots,S \\ H^s \geqslant 1}} b^s(n_i)$.

For each Stack s such that $H^s \geqslant 1$, consider the containers in this stack $(c_i^s)_{i=1,\dots,H^s}$. Since all containers labeled $K_{w_{min}}$, i.e., from batch w_{min}, are known in n_i, we can write

$$\begin{aligned} b^s(n_i) &= \sum_{h=1,\dots,H^s} \mathbb{P}\left[c_h^s \textit{ is blocking in } n_i\right] \\ &= \sum_{\substack{h=1,\dots,H^s \\ c_h^s = K_{w_{min}}}} \mathbb{1}\left\{c_h^s \textit{ is blocking in } n_i\right\} + \sum_{\substack{h=1,\dots,H^s \\ c_h^s \neq K_{w_{min}}}} \mathbb{P}\left[c_h^s \textit{ is blocking in } n_i\right]. \end{aligned}$$

Fix $h \in \{1, \dots, H^s\}$ and $c_h^s \neq K_{w_{min}}$, then the proof of Lemma 5 uses the fact that $\mathbb{P}\left[c_h^s \textit{ is blocking in } n_i\right] = 1 - \frac{\mathbb{1}\left\{c_h^s = \min\limits_{i=1,\dots,h} \{c_i^s\}\right\}}{\sum_{i=1}^{h} \mathbb{1}\{c_h^s = c_i^s\}}$. Finally, it is clear that $0 \leqslant \sum\limits_{\substack{h=1,\dots,H^s \\ c_h^s = K_{w_{min}}}} \mathbb{1}\left\{c_h^s \textit{ is blocking in } n_i\right\} \leqslant C^s_{w_{min}}$. Therefore, we can get the corresponding formulas. As a final remark, note that each of these bounds is tight. Indeed, consider the offspring of n, in which all containers in batch w_{min} are in the decreasing (resp. increasing) order of retrieval from top to bottom, then this offspring has no (resp. $C^s_{w_{min}}$) blocking container(s). □

B.4 Computational Experiments Tables

			Lower bounds			*PBFS*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	8	1.64	1.66	1.66	1.70	**1.70**	**1.70**	**1.70**	1.82	2.34
	4	10	2.88	2.96	2.99	3.11	**3.11**	3.13	3.14	3.51	4.62
	5	13	4.61	4.88	5.01	5.32	5.40	**5.38**	5.57	6.17	8.00
	6	15	6.28	6.64	7.06	7.59	7.85	**7.81**	8.09	9.41	12.35
6	3	9	1.68	1.69	1.69	1.74	1.76	**1.74**	**1.74**	1.84	2.43
	4	12	3.54	3.61	3.63	3.68	3.69	**3.68**	3.68	4.11	5.59
	5	15	5.37	5.57	5.68	5.91	5.97	**5.94**	6.01	7.21	9.55
	6	18	7.19	7.52	7.68	8.23	8.38	**8.29**	8.63	10.05	13.53
7	3	11	2.82	2.86	2.88	2.88	**2.88**	**2.88**	2.89	2.96	4.12
	4	14	3.97	4.06	4.10	4.16	**4.17**	**4.17**	4.20	4.65	6.47
	5	18	6.49	6.65	6.74	6.97	7.05	**7.00**	7.07	8.46	11.27
	6	21	8.82	9.21	9.51	-	10.40	**10.35**	10.76	12.47	17.69
8	3	12	2.29	2.30	2.30	2.31	**2.31**	**2.31**	**2.31**	2.43	3.23
	4	16	4.68	4.73	4.75	4.82	**4.83**	**4.83**	**4.83**	5.41	7.48
	5	20	7.20	7.42	7.54	7.85	7.96	**7.93**	8.06	9.32	13.44
	6	24	9.52	9.85	10.09	-	11.10	**10.99**	11.34	13.29	19.28
9	3	14	2.98	2.98	2.98	3.00	**3.00**	3.01	**3.00**	3.19	4.54
	4	18	5.63	5.71	5.71	5.73	**5.73**	**5.73**	**5.73**	6.52	9.29
	5	23	8.58	8.69	8.77	-	9.05	**9.02**	9.12	11.16	15.57
	6	27	10.38	10.78	10.98	-	11.59	**11.58**	11.76	14.62	20.93
10	3	15	3.18	3.18	3.18	3.19	**3.19**	3.20	3.20	3.27	4.75
	4	20	6.20	6.23	6.23	6.28	6.30	**6.28**	**6.28**	6.98	10.41
	5	25	9.10	9.37	9.39	-	**9.60**	**9.60**	9.73	11.38	16.64
	6	30	11.91	12.28	12.44	-	13.01	**12.92**	13.15	15.93	23.40

Table B.1: Results of experiment 1: Performance of *PBFS*, heuristics, and tightness of lower bounds for a fill rate of 50 percent in the batch model, in the case of small batches. Bold numbers highlight the best heuristic for a given problem size.

			Lower bounds			*PBFS*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	10	2.83	2.97	3.01	3.08	**3.08**	**3.08**	3.11	3.33	4.16
	4	13	4.69	4.97	5.09	5.58	5.69	**5.64**	5.75	6.50	8.22
	5	17	7.58	8.52	8.92	-	**10.44**	10.48	11.04	12.23	15.27
	6	20	9.69	10.79	11.46	-	**14.53**	14.65	15.77	18.47	22.93
6	3	12	3.60	3.70	3.72	3.89	**3.89**	3.90	3.90	4.32	5.54
	4	16	6.20	6.59	6.78	7.28	**7.41**	7.45	7.61	8.55	11.29
	5	20	8.28	8.85	9.16	-	10.39	**10.38**	10.80	12.85	16.43
	6	24	11.67	12.22	12.54	-	15.19	**15.17**	16.14	19.33	25.03
7	3	14	3.85	3.89	3.91	3.97	**3.98**	**3.98**	4.02	4.40	6.05
	4	19	6.25	6.60	6.86	7.29	7.37	**7.36**	7.54	8.89	11.60
	5	23	9.72	10.24	10.55	-	11.75	**11.71**	12.30	14.92	19.65
	6	28	13.52	14.41	14.93	-	17.72	**17.63**	18.81	23.10	30.78
8	3	12	4.47	4.57	4.61	4.66	**4.66**	**4.66**	4.68	5.14	6.94
	4	21	7.62	7.85	7.98	8.29	**8.33**	8.35	8.43	9.76	13.26
	5	27	11.61	12.08	12.52	-	13.56	**13.47**	14.10	17.15	23.11
	6	32	15.60	16.39	16.78	-	**19.28**	19.51	20.85	25.89	34.66
9	3	18	4.81	4.96	4.99	5.10	**5.10**	5.12	5.14	5.66	7.81
	4	24	8.98	9.18	9.30	9.58	9.63	**9.61**	9.76	11.66	16.00
	5	30	13.16	13.90	14.29	-	**15.65**	15.79	16.75	20.03	27.41
	6	36	16.77	17.36	17.83	-	**20.38**	20.40	21.86	28.12	38.12
10	3	20	5.21	5.21	5.21	5.27	**5.28**	**5.28**	**5.28**	5.79	7.86
	4	27	9.18	9.54	9.71	-	**10.27**	10.29	10.37	12.16	16.91
	5	34	14.46	14.88	15.16	-	**16.13**	16.19	16.69	21.06	29.03
	6	40	19.55	20.24	20.66	-	23.33	**23.20**	24.46	32.11	44.07

Table B.2: Results of experiment 1: Performance of *PBFS*, heuristics, and tightness of lower bounds for a fill rate of 67 percent in the batch model, in the case of small batches. Bold numbers highlight the best heuristic for a given problem size.

			Lower bounds			*PBFSA*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	8	1.68	1.68	1.68	1.76	1.77	**1.76**	**1.76**	1.87	2.38
	4	10	2.96	3.02	3.05	3.33	**3.36**	3.37	3.38	3.67	4.85
	5	13	4.58	4.82	4.93	5.50	5.69	**5.65**	5.74	6.33	8.22
	6	15	6.31	6.72	6.96	7.81	8.28	**8.03**	8.35	9.49	12.40
6	3	9	1.66	1.67	1.67	1.73	1.75	**1.74**	**1.74**	1.82	2.43
	4	12	3.61	3.71	3.73	3.93	**3.99**	**3.99**	4.00	4.34	5.82
	5	15	5.38	5.57	5.64	6.11	**6.23**	**6.23**	6.31	7.16	9.65
	6	18	7.01	7.26	7.44	-	8.49	**8.36**	8.61	9.92	13.45
7	3	11	2.76	2.79	2.79	2.85	2.84	2.84	**2.83**	2.95	4.06
	4	14	4.02	4.12	4.15	4.24	4.31	**4.29**	4.31	4.73	6.52
	5	18	6.29	6.39	6.43	6.77	7.00	**6.92**	7.01	8.20	11.08
	6	21	8.69	9.12	9.35	-	10.60	**10.52**	10.91	12.61	17.79
8	3	12	2.30	2.31	2.31	2.31	**2.31**	2.32	2.32	2.37	3.19
	4	16	4.61	4.62	4.63	4.71	**4.74**	**4.74**	4.75	5.25	7.40
	5	20	7.31	7.46	7.52	-	**8.01**	**8.01**	8.09	9.33	13.25
	6	24	9.65	9.95	10.12	-	**11.37**	**11.37**	11.67	13.44	19.51
9	3	14	2.93	2.93	2.93	2.95	**2.96**	**2.96**	**2.96**	3.15	4.48
	4	18	5.56	5.58	5.59	5.69	5.74	**5.70**	**5.70**	6.33	9.07
	5	23	8.49	8.64	8.73	-	9.16	**9.12**	9.16	10.99	15.39
	6	27	10.38	10.69	10.90	-	11.77	**11.75**	11.95	14.65	20.95
10	3	15	3.15	3.16	3.16	3.15	**3.17**	**3.17**	**3.17**	3.25	4.72
	4	20	6180	6.20	6.21	6.28	6.35	**6.34**	**6.34**	6.92	10.27
	5	25	9.13	9.31	9.36	-	9.66	**9.63**	9.68	11.44	16.73
	6	30	12.09	12.35	12.51	-	13.38	**13.23**	13.42	16.33	23.80

Table B.3: Results of experiment 2: Performance of *PBFSA*, heuristics, and tightness of lower bounds for a fill rate of 50 percent in the batch model with larger batches. Bold numbers highlight the best heuristic for a given problem size.

			Lower bounds			*PBFSA*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	10	2.78	2.87	2.90	3.07	**3.08**	**3.08**	3.09	3.36	4.18
	4	13	4.71	4.92	5.00	5.70	5.81	**5.80**	5.89	6.60	8.23
	5	17	7.53	8.17	8.44	-	10.38	**10.32**	10.68	11.96	14.9815
	6	20	9.69	10.56	11.12	-	15.00	**14.91**	16.01	18.22	22.79
6	3	12	3.56	3.63	3.64	3.90	**3.90**	3.91	3.92	4.28	5.55
	4	16	6.12	6.48	6.61	7.17	7.36	**7.41**	7.48	8.44	11.02
	5	20	8.33	8.72	8.90	-	10.36	**10.30**	10.57	12.54	16.15
	6	24	11.77	12.41	12.80	-	15.94	**15.91**	16.86	19.83	25.45
7	3	14	3.95	4.01	4.02	4.14	4.17	**4.15**	**4.15**	4.56	6.11
	4	19	6.27	6.56	6.77	-	7.36	**7.35**	7.52	8.66	11.46
	5	23	9.81	10.27	10.54	-	**12.08**	12.10	12.49	14.78	19.67
	6	28	13.64	14.47	14.91	-	18.36	**18.26**	19.32	23.10	31.03
8	3	12	4.65	4.74	4.76	4.85	4.86	**4.85**	4.88	5.34	7.14
	4	21	7.58	7.86	7.99	-	**8.38**	8.42	8.50	9.82	13.20
	5	27	11.46	11.98	12.28	-	13.73	**13.60**	14.11	17.00	22.84
	6	32	15.45	16.28	16.72	-	19.94	**19.83**	21.21	25.73	34.70
9	3	18	4.85	4.98	5.02	5.14	**5.17**	5.20	5.20	5.67	7.77
	4	24	8.82	9.00	9.11	-	9.66	**9.60**	9.72	11.52	15.70
	5	30	13.15	13.84	14.18	-	**15.91**	15.96	16.82	20.16	27.35
	6	36	16.85	17.39	17.80	-	20.99	**20.83**	21.97	28.05	38.04
10	3	20	5.19	5.21	5.22	5.31	**5.31**	**5.31**	**5.31**	5.79	7.92
	4	27	9.40	9.66	9.82	-	10.47	**10.46**	10.54	12.25	16.96
	5	34	14.44	14.83	15.07	-	**16.29**	**16.29**	16.62	21.12	28.86
	6	40	19.49	20.24	20.66	-	23.83	**23.65**	24.96	32.04	44.12

Table B.4: Results of experiment 2: Performance of *PBFSA*, heuristics, and tightness of lower bounds for a fill rate of 67 percent in the batch model with larger batches. Bold numbers highlight the best heuristic for a given problem size.

			Lower bounds			*PBFS*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	8	1.64	1.66	1.66	1.70	**1.70**	**1.70**	1.71 (1.71)	1.82	2.34 (2.34)
	4	10	2.88	2.96	2.99	3.11	**3.11**	3.13	3.14 (3.20)	3.51	4.62 (4.62)
	5	13	4.61	4.88	5.01	5.32	**5.38**	**5.38**	5.57 (5.58)	6.16	8.00 (8.00)
	6	15	6.28	6.64	7.06	7.59	7.85	**7.80**	8.08 (8.29)	9.41	12.36 (12.35)
6	3	9	1.68	1.69	1.69	1.74	1.76	**1.74**	**1.74** (1.75)	1.84	2.43 (2.43)
	4	12	3.54	3.61	3.63	3.68	3.69	**3.68**	**3.68** (3.75)	4.11	5.59 (5.59)
	5	15	5.37	5.57	5.68	5.91	5.96	**5.94**	6.00 (6.18)	7.21	9.54 (9.54)
	6	18	7.19	7.52	7.68	8.23	8.38	**8.29**	8.62 (8.77)	10.05	13.53 (13.53)
7	3	11	2.82	2.86	2.88	2.88	**2.88**	**2.88**	2.89 (2.88)	2.96	4.11 (4.11)
	4	14	3.97	4.06	4.1	4.16	**4.17**	**4.17**	4.21 (4.20*)	4.66	6.47 (6.03*)
	5	18	6.49	6.65	6.74	6.97	7.04	**7.00**	7.07 (7.18)	8.45	11.27 (11.27)
	6	21	8.82	9.21	9.51	-	10.40	**10.35**	10.76 (10.98)	12.46	17.69 (17.69)
8	3	12	2.29	2.3	2.3	2.31	**2.31**	**2.31**	**2.31** (2.32)	2.43	3.23 (3.23)
	4	16	4.68	4.73	4.75	4.82	**4.83**	**4.83**	**4.83** (4.88)	5.41	7.49 (7.49)
	5	20	7.20	7.42	7.54	7.85	7.97	**7.94**	8.07 (8.27)	9.32	13.44 (13.45)
	6	24	9.52	9.85	10.09	-	11.10	**10.98**	11.34 (11.61)	13.29	19.29 (19.29)
9	3	14	2.98	2.98	2.98	3.00	**3.00**	**3.00**	**3.00** (3.00)	3.19	4.54 (4.54)
	4	18	5.63	5.71	5.71	5.73	**5.73**	**5.73**	**5.73** (5.80)	6.52	9.29 (9.29)
	5	23	8.58	8.69	8.77	-	9.05	**9.02**	9.12 (9.36)	11.16	15.56 (15.57)
	6	27	10.38	10.78	10.98	-	11.59	**11.58**	11.76 (12.09)	14.62	20.94 (20.93)
10	3	15	3.18	3.18	3.18	3.19	**3.19**	3.20	3.20 (3.20)	3.27	4.75 (4.75)
	4	20	6.20	6.23	6.23	6.28	6.30	**6.27**	6.28 (6.33)	6.98	10.41 (10.41)
	5	25	9.10	9.37	9.39	-	9.61	**9.60**	9.73 (9.80)	11.38	16.64 (16.63)
	6	30	11.91	12.28	12.44	-	13.01	**12.92**	13.15 (13.51)	15.92	23.41 (23.41)

Table B.5: Results of experiment 3: Performance of heuristics and tightness of lower bounds for a fill rate of 50 percent in the online model with small batches. Bold numbers highlight the best heuristic for a given problem size. Numbers in parentheses are taken from [43].

			Lower bounds			*PBFS*	Heuristics				
S	T	C	b	b_1	b_2		EG	EM	ERI	L	Rand.
5	3	10	2.83	2.97	3.01	3.08	**3.08**	**3.08**	3.12 (3.10)	3.33	4.16 (4.16)
	4	13	4.69	4.97	5.09	5.58	5.68	**5.64**	5.75 (5.80)	6.50	8.22 (8.22)
	5	17	7.58	8.52	8.92	-	**10.45**	10.48	11.04 (11.15)	12.24	15.28 (15.28)
	6	20	9.69	10.79	11.46	-	**14.53**	14.65	15.77 (16.14)	18.46	22.93 (22.93)
6	3	12	3.6	3.7	3.72	3.89	**3.89**	3.90	3.90 (3.92)	4.32	5.53 (5.53)
	4	16	6.2	6.59	6.78	7.28	**7.41**	7.45	7.61 (7.68)	8.54	11.29 (11.28)
	5	20	8.28	8.85	9.16	-	**10.38**	**10.38**	10.80 (10.97)	12.85	16.42 (16.42)
	6	24	11.67	12.22	12.54	-	**15.17**	**15.17**	16.14 (16.65)	19.33	25.04 (25.03)
7	3	14	3.85	3.89	3.91	3.97	**3.98**	**3.98**	4.02 (4.01)	4.40	6.05 (6.05)
	4	19	6.25	6.6	6.86	7.29	7.37	**7.36**	7.54 (7.68)	8.89	11.60 (11.61)
	5	23	9.72	10.24	10.55	-	11.76	**11.71**	12.30 (12.64)	14.92	19.66 (19.65)
	6	28	13.52	14.41	14.93	-	17.70	**17.64**	18.82 (19.49)	23.10	30.77 (30.79)
8	3	12	4.47	4.57	4.61	4.66	**4.65**	4.66	4.68 (4.7)	5.14	6.94 (6.94)
	4	21	7.62	7.85	7.98	8.29	**8.32**	8.35	8.43 (8.5)	9.75	13.26 (13.25)
	5	27	11.61	12.08	12.52	-	13.56	**13.47**	14.10 (14.44)	17.14	23.11 (23.12)
	6	32	15.6	16.39	16.78	-	**19.27**	19.51	20.85 (21.72)	25.89	34.64 (34.63)
9	3	18	4.81	4.96	4.99	5.10	**5.10**	5.12	5.14 (5.19)	5.66	7.80 (7.80)
	4	24	8.98	9.18	9.3	9.58	9.63	**9.61**	9.76 (9.92)	11.66	16.01 (16.00)
	5	30	13.16	13.9	14.29	-	**15.65**	15.79	16.75 (16.97)	20.03	27.38 (27.39)
	6	36	16.77	17.36	17.83	-	**20.38**	20.40	21.87 (22.73)	28.13	38.11 (38.14)
10	3	20	5.21	5.21	5.21	5.27	**5.28**	**5.28**	**5.28** (5.30)	5.79	7.85 (7.86)
	4	27	9.18	9.54	9.71	-	**10.27**	10.29	10.37 (10.50)	12.15	16.92 (16.91)
	5	34	14.46	14.88	15.16	-	**16.13**	16.19	16.69 (17.23)	21.07	29.03 (29.03)
	6	40	19.55	20.24	20.66	-	23.33	**23.20**	24.46 (25.58)	32.11	44.08 (44.07)

Table B.6: Results of experiment 3: Performance of heuristics and tightness of lower bounds for a fill rate of 67 percent in the online model with small batches. Bold numbers highlight the best heuristic for a given problem size. Numbers in parentheses are taken from [43].

Appendix C

Appendix on the Yard Crane Scheduling Problem with Relocations

C.1 Notations Summary

X: the number of rows of the block. Typical values range from 6 to 13.

Y: the number of bays of the block. Typical values range from 10 to 40.

Z: the number of tiers of the block, also the maximum number of containers in a given stack. Typical values range from 3 to 6.

$s = (s_x, s_y)$: a stack of the block identified by its two coordinates.

$\mathcal{S}_B$: the set of stacks in the block.

M: total number of I/O points (M_1, number of I/O points on seaside or internal yard side; M_2, number of I/O points on landside or external yard side). Typical values are of the order of number of bays for Asian and double-sided styles and of the order of rows for European style.

$\mathcal{S}_I$: the set of I/O points or "artificial" stacks.

$\mathcal{S}$: the set of all stacks.

z_s^i: initial number of containers stored in stack $s \in \mathcal{S}_B$. We have $z_s^i \in \{0, \ldots, Z\}$.

s^i: initial position of the YC ($s^i \in \mathcal{S}$).

$\left(v^{x,E}, v^{x,L}\right)$: YC trolley speed without and with load (both equal to 0.50 containers/s from Table C.1).

$\left(v^{y,E}, v^{y,L}\right)$: YC gantry speed without and with load (0.37 and 0.20 containers/s).

$\left(v^{z,E}, v^{z,L}\right)$: YC speed to lower and hoist the spreader (0.39 and 0.20 containers/s).

v^z: harmonic mean of $v^{z,E}$ and $v^{z,L}$ (0.26 containers/s).

t^h: handling time (or stabilization time) to pick up or set down a container on a stack (20 s).

$t_{sr}^{(E)}$: time for an empty drive of the YC from stack s to stack r.

$t_{sr}^{(L)}$: time for a loaded drive of the YC from stack s to stack r.

$t^{(H)}(z)$: time for lifting/setting down a container from/onto a stack on tier $z \in \{1, \ldots, Z\}$. The I/O points is equivalent to set on the ground (i.e., $z = 1$)

N: the number of requests (or productive moves). Typical values range from 1 to 15. Requests are indexed based on their arrival order.

$\mathcal{N}_s$: the indices corresponding to storage requests ($\mathcal{N}_s \subset \{1, \ldots, N\}$).

$\mathcal{N}_r$: the indices corresponding to retrieval requests ($\mathcal{N}_r \subset \{1, \ldots, N\}$).

(δ_n^-, δ_n^+): flexibility of request n. Request n can be served between the $n - \delta_n^-$-th request and the $n + \delta_n^+$-th request.

z_n: the tier at which container n ($\in \mathcal{S}_r$) is stored in stack s ($\in L_n$). Note that $z_n \in \{1, \ldots, z_s^i\}$.

$\mathcal{N}_u$: the indices corresponding to unproductive requests implied by retrieval requests.

$\overline{N}$: the total number of requests (including relocations) to perform by the YC to fulfill all N productive requests.

L_n: the set of stacks in which the container $n \in \{1, \ldots, \overline{N}\}$ can be picked up by the crane.

E_n: the set of stacks onto which container $n \in \{1, \ldots, \overline{N}\}$ can be put down.

b_n: container directly blocking $n \in \mathcal{N}_r \cup \mathcal{N}_u$. If n is on the top of its stack, then $b_n = 0$.

$\mathcal{S}^{(L)}$: the set of starting stacks for loaded drives of the YC.

$\mathcal{S}^{(E)}$: the set of starting stacks for empty drives of the YC.

$\mathcal{S}_R$: the set of stacks of the block where there is at least one container that needs to be retrieved.

$\widetilde{z}_r$: the number of containers in stack $r \in \mathcal{S}_B$ after all containers have been retrieved and none has been stored or relocated.

m_r: the lowest container to be retrieved in $r \in \mathcal{S}_R$.

γ: the weight on the cost-to-go.

α_z: the expected number of blocking containers in a stack with z containers.

C.2 Technical Proofs

Lemma 11. *Let $n \in \mathcal{N}_s \cup \mathcal{N}_u$. Let $v(n)$ be the tier at which container n is stored or relocated when performing request n and z_r^f the number of containers in stack r after*

Variable	Value
Trolley speed without load of the YC	1.17 m/s
Trolley speed with load of the YC	1.17 m/s
Gantry speed without load of the YC	2.17 m/s
Gantry speed with load of the YC	1.17 m/s
Hoisting speed without load of the YC	0.93 m/s
Hoisting speed with load of the YC	0.47 m/s
Container width	2.35 m
Container length	5.90 m
Container height	2.39 m
Time to handle and stabilize container	20 s

Table C.1: Inputs of the simulation study (yard speed from Liebherr.com and TEU size from dsv.com). Assumptions: No acceleration is considered. All containers are 20 feet long and dry. Note that these values are similar to [28]. No separating space between containers is considered.

performing all $\overline{N}$ requests, then we have

$$\sum_{n\in\mathcal{N}_s\cup\mathcal{N}_u} v(n) = \frac{1}{2}\sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} z_r^f\left(z_r^f+1\right) - \frac{1}{2}\sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \widetilde{z}_r\left(\widetilde{z}_r+1\right).$$

Proof. First, note that each container can only be moved once. Therefore, the tier at which container n is stored or relocated when performing request n is the same as the tier at which container n is stored after all $\overline{N}$ requests are performed.

Consider a given stack $r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B$. If $z_r^f = \widetilde{z}_r$, then no container was stored or relocated to stack r. If $z_r^f > \widetilde{z}_r$, then there are $z_r^f - \widetilde{z}_r$ containers that got stored or relocated to stack r, each of which corresponds to a unique $n \in \mathcal{N}_s \cup \mathcal{N}_u$. The tiers of stack r at which these containers got stacked range from $\widetilde{z}_r + 1, \ldots, z_r^f$.

By summing this observation for all stacks $r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B$, we get the tiers of all containers $n \in \mathcal{N}_s \cup \mathcal{N}_u$. Indeed, no container can be stored or relocated to a stack

$r' \in \mathcal{S}_B \setminus \mathcal{S}^{(E)}$. Thus:

$$\begin{aligned}\sum_{n\in\mathcal{N}_s\cup\mathcal{N}_u} v(n) &= \sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \sum_{z=\widetilde{z}_r+1}^{z_r^f} z \\ &= \sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \left(z_r^f - \widetilde{z}_r\right) \frac{\left(z_r^f + \widetilde{z}_r + 1\right)}{2} \\ &= \frac{1}{2} \sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} z_r^f \left(z_r^f + 1\right) - \frac{1}{2} \sum_{r\in\mathcal{S}^{(E)}\cap\mathcal{S}_B} \widetilde{z}_r \left(\widetilde{z}_r + 1\right).\end{aligned}$$

□

Lemma 12. *Let γ verify Condition (A), $z, z_1, z_2 \in \{0, \ldots, Z\}$ such that $z_2 < z < z_1$, then we have*

$$\frac{z - z_2}{z_1 - z_2}\beta_{z_1} + \frac{z_1 - z}{z_1 - z_2}\beta_{z_2} > \beta_z.$$

Proof. First, we show that

$$\beta_{z+1} + \beta_{z-1} > 2\beta_z. \tag{C.1}$$

By definition of β_z, this inequality is equivalent to

$$\frac{\gamma}{z(z+1)} - \frac{2}{v^z} > 0.$$

Since $z < z_1 \leqslant Z$, we have $z \leqslant Z - 1$. In addition, γ verifies Condition (A), i.e., $\gamma > \dfrac{2Z(Z-1)}{v^z}$. Thus, we get

$$\frac{\gamma}{z(z+1)} - \frac{2}{v^z} \geqslant \frac{\gamma}{(Z-1)Z} - \frac{2}{v^z} > \frac{2}{v^z} - \frac{2}{v^z} = 0,$$

which completes the proof of Equation (C.1). Using Equation (C.1) repeatedly, we have

$$\beta_{z_1} + (z_1 - z)\,\beta_{z-1} > (z_1 - z + 1)\,\beta_z \text{ and } (z - z_2)\,\beta_{z+1} + \beta_{z_2} > (z - z_2 + 1)\,\beta_z. \tag{C.2}$$

Consequently, we have

$$\begin{aligned}\frac{z-z_2}{z_1-z_2}\beta_{z_1}+\frac{z_1-z}{z_1-z_2}\beta_{z_2} &> \frac{(z-z_2)(z_1-z+1)}{z_1-z_2}\beta_z-\frac{(z-z_2)(z_1-z)}{z_1-z_2}\beta_{z-1}\\ &\quad+\frac{(z_1-z)(z-z_2+1)}{z_1-z_2}\beta_z-\frac{(z_1-z)(z-z_2)}{z_1-z_2}\beta_{z+1}\\ &= \beta_z-\frac{(z_1-z)(z-z_2)}{z_1-z_2}(\beta_{z+1}+\beta_{z-1}-2\beta_z)\\ &> \beta_z,\end{aligned}$$

where the first inequality holds thanks to Equation (C.2) and the second holds with Equation (C.1) and $z_2 < z < z_1$. This concludes the proof of Lemma 12. □

Lemma 13. *Let $w \in \mathcal{W} \cap \{0,1\}$. Let $\Pi(w)$ be the optimization problem defined as*

$$\min_{D\in\mathcal{L}(w)\cap\{0,1\}} \left(c^T D\right),$$

and $\overline{\Pi}(w)$ the optimization problem defined as

$$\min_{\overline{D}\in\overline{\mathcal{L}}(w)\cap\{0,1\}} \left(\overline{c}^T \overline{D}\right),$$

then $\Pi(w)$ and $\overline{\Pi}(w)$ are equivalent problems.

Proof. To prove the equivalence between $\Pi(w)$ and $\overline{\Pi}(w)$, we show that $\Pi(w)$ has several implied equalities/inequalities. Using these implied constraints, we can reduce the number of variables and constraints to obtain the formulation of $\overline{\Pi}(w)$. We prove the following implied inequalities in this order:

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \begin{cases} \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0, \\ \forall r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k rk} = 0. \end{cases} \tag{C.3}$$

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \sum_{r\in\overline{E}_{\nu_k}} d^{(L)}_{\sigma_k rk} = 1. \tag{C.4}$$

$$\begin{cases} d^i_{\sigma_1} = 1, \\ \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\},\ d^i_s = 0. \end{cases} \tag{C.5}$$

$$\forall\ k \in \{2, \ldots, \overline{N}\},\ \forall\ r \in \mathcal{S}^{(E)},\ \forall s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ d^{(E)}_{rsk} = 0. \tag{C.6}$$

$$\forall\ k \in \{2, \ldots, \overline{N}\},\ \forall\ r \in \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} - d^{(L)}_{\sigma_{k-1},r,k-1} = 0. \tag{C.7}$$

$$\forall\ k \in \{2, \ldots, \overline{N}\},\ \forall\ r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} = 0. \tag{C.8}$$

$$\forall r \in \overline{\mathcal{S}}_B,\ \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k} = \widetilde{z}_r. \tag{C.9}$$

$$\forall r \in \overline{\mathcal{S}}_B,\ \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1, \tag{C.10}$$

$$\forall r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B,\ \begin{cases} f_{r\widetilde{z}_r} = 1, \\ \forall\ z \in \{\widetilde{z}_r + 1, \ldots, Z\}, f_{rz} = 0. \end{cases} \tag{C.11}$$

Proof of implied Constraint (C.3). We first show that

$$\begin{gathered} \forall\ k \in \{1, \ldots, \overline{N}\}, \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0, \\ \forall\ k \in \{1, \ldots, \overline{N}\},\ \forall r \in \mathcal{S}^{(E)} \setminus E_{\nu_k},\ d^{(L)}_{\sigma_k r k} = 0. \end{gathered} \tag{C.12}$$

Let $k \in \{1, \ldots, \overline{N}\}$. By definition we have $w_{\nu_k \sigma_k k} = 1$, so Constraint (5.13) for $n = \nu_k$ and $s = \sigma_k$ implies that

$$0 \leqslant \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k r k} - w_{\nu_k \sigma_k k} = \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k r k} - 1.$$

We combine this inequality with Constraint (5.8) which implies that

$$\begin{aligned} 0 &= \sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} - 1 \\ &= \sum_{\substack{s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\} \\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} + \sum_{r \in \mathcal{S}^{(E)} \setminus E_{\nu_k}} d^{(L)}_{\sigma_k r k} + \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k r k} - 1 \\ &\geqslant \sum_{\substack{s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\} \\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} + \sum_{r \in \mathcal{S}^{(E)} \setminus E_{\nu_k}} d^{(L)}_{\sigma_k r k}, \end{aligned}$$

Since $d^{(L)}_{srk}$ are non-negative, this proves Equation (C.12). We further improve this constraint by showing that

$$\forall\ k \in \{1, \dots, \overline{N}\}, \forall\ r \in E_{\nu_k} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k r k} = 0. \tag{C.13}$$

Let $k \in \{1, \dots, \overline{N}\}$ and $r \in E_{\nu_k} \setminus \overline{E}_{\nu_k}$. First, using Equation (C.12), we have $\sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} = d^{(L)}_{\sigma_k r k}$. Moreover, by definition of $\overline{E}_{\nu_k}$, we must have $r \in \mathcal{S}_R$ and $\forall\ k' \in \{l, \dots, k-1\}$, $\nu_{k'} \neq m_r$. Since $w \in \mathcal{W}$, it implies that $\forall\ k' \in \{1, \dots, k-1\}$, $w_{m_r r k'} = 0$, thus $\sum_{k' \in \{1,\dots,k-1\}} w_{m_r r k'} = 0$. By using both these observations in Constraint (5.14), we have

$$0 \geqslant \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k' \in \{1,\dots,k-1\}} w_{m_r r k'} = d^{(L)}_{\sigma_k r k}.$$

As $d^{(L)}_{\sigma_k r k}$ are non-negative, this proves Equation (C.13). Combined with Equation (C.12), this proves implied Constraint (C.3).

Proof of implied Constraint (C.4). Using implied Constraint (C.3) together with Constraint (5.8), we directly get implied Constraint (C.4).

Proof of implied Constraint (C.5). We have

$$d^i_{\sigma_1} = \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_1 r 1} = \sum_{r \in \overline{E}_{\nu_1}} d^{(L)}_{\sigma_1 r 1} = 1,$$

where the first equality comes from Constraint (5.9a) for $s = \sigma_1$, the second from implied Constraint (C.3) and the last from implied Constraint (C.4) for $k = 1$. By combining this latter fact with Constraint (5.7a), we get

$$0 = \sum_{s \in \mathcal{S}^{(L)}} d^i_s - 1 = \sum_{s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\}} d^i_s + d^i_{\sigma_1} - 1 = \sum_{s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\}} d^i_s,$$

thus, since d^i_s are non-negative, we get implied Constraint (C.5).

Proof of implied Constraint (C.6). Let $k \in \{2, \ldots, \overline{N}\}$, we first have

$$\sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{r\sigma_k k} = \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_k r k} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} = 1,$$

where the first equality is Constraint (5.9b) for $s = \sigma_k$, the second equality comes from implied Constraint (C.3) and the last from implied Constraint (C.4). Based on this observation, we use Constraint (5.7b) to get

$$0 = \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)}}} d^{(E)}_{rsk} - 1 = \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\}}} d^{(E)}_{rsk} + \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{r\sigma_k k} - 1 = \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\}}} d^{(E)}_{rsk}$$

which by non-negativity of $d^{(E)}_{rsk}$ proves Equation (C.6).

Proof of implied Constraint (C.7). Let $k \in \{2, \ldots, \overline{N}\}$ and $r \in \overline{E}_{\nu_{k-1}}$, we have

$$0 = \sum_{s \in \mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = d^{(E)}_{r\sigma_k k} - d^{(L)}_{\sigma_{k-1},r,k-1},$$

which proves implied Constraint (C.7). The first equality is Constraint (5.10) since $r \in \overline{E}_{\nu_{k-1}} \subset E_{\nu_{k-1}} \subset \mathcal{S}^{(E)}$. The second equality results from both implied Constraints (C.6) and (C.3).

Proof of implied Constraint (C.8). Let $k \in \{2, \ldots, \overline{N}\}$ and $r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}}$. Using both implied Constraints (C.7) and (C.3), we have

$$d^{(E)}_{r\sigma_k k} = d^{(L)}_{\sigma_{k-1},r,k-1} = 0,$$

proving implied Constraint (C.8).

Proof of implied Constraint (C.9). Let $r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B$. Using implied Constraint (C.3) and by definition of $\overline{K}_r$, we note that

$$\sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1,\dots,\overline{N}\}}} d^{(L)}_{srk} = \sum_{k \in \{1,\dots,\overline{N}\}} d^{(L)}_{\sigma_k rk} = \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k rk}.$$

By replacing this expression in Constraint (5.12), we get

$$\forall\ r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \sum_{z \in \{\widetilde{z}_r,\dots,Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k rk} = \widetilde{z}_r. \tag{C.14}$$

Since $\overline{\mathcal{S}}_B \subset \mathcal{S}^{(E)} \cap \mathcal{S}_B$, this proves implied Constraint (C.9).

Proof of implied Constraint (C.10). This implied constraint is directly proven by Constraint (5.11) since $\overline{S}_B \subset \mathcal{S}^{(E)} \cap \mathcal{S}_B$.

Proof of implied Constraint (C.11). Let $r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B$. By definition it implies that $r \in \mathcal{S}^{(E)}$ and $r \notin \bigcup_{k \in \{1,\dots,\overline{N}\}} \overline{E}_{\nu_k}$, i.e., $\forall k \in \{1,\dots,\overline{N}\}$, $r \notin \overline{E}_{\nu_k}$. In conclusion, $\forall k \in \{1,\dots,\overline{N}\}$ $r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_k}$. Using this fact with implied Constraint (C.3) we have $\forall k \in \{1,\dots,\overline{N}\}$, $d^{(L)}_{\sigma_k rk} = 0$, thus

$$\sum_{k \in \{1,\dots,\overline{N}\}} d^{(L)}_{\sigma_k rk} = 0.$$

Consequently, we have

$$\begin{aligned}
0 &= \widetilde{z}_r - \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} + \sum_{k \in \{1, \ldots, \overline{N}\}} d^{(L)}_{\sigma_k r k} \\
&= \widetilde{z}_r - \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} \\
&= \widetilde{z}_r (1 - f_{r\widetilde{z}_r}) + \sum_{z \in \{\widetilde{z}_r + 1, \ldots, Z\}} z f_{rz} \\
&\geqslant \sum_{z \in \{\widetilde{z}_r + 1, \ldots, Z\}} z f_{rz} \\
&\geqslant \sum_{z \in \{\widetilde{z}_r + 1, \ldots, Z\}} f_{rz},
\end{aligned}$$

and since f_{rz} are non-negative, this proves that $\forall\ z \in \{\widetilde{z}_r + 1, \ldots, Z\}$, $f_{rz} = 0$. We note that the first equality comes from Constraint (5.12), the second from the previous observation, the first inequality from $f_{r\widetilde{z}_r} \leqslant 1$ and the second inequality from $z \geqslant 1$ for $z \in \{\widetilde{z}_r + 1, \ldots, Z\}$. Finally, using this fact and Constraint (5.11), we have

$$1 = \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = f_{r\widetilde{z}_r},$$

which proves implied Constraint (C.11).

Redundancy of original constraints. We now show that Constraints (5.7)-(5.14) are redundant with implied Constraints (C.3)-(C.11).

Constraint (5.7a) is redundant with implied Constraint (C.5):

$$\sum_{s \in S^{(L)}} d^i_s = d^i_{\sigma_1} = 1.$$

Constraint (5.7b) is redundant with implied Constraints (C.6), (C.8), (C.7) and (C.5):

$$\forall\ k \in \{2, \ldots, \overline{N}\},\ \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)}}} d^{(E)}_{rsk} = \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(E)}_{r\sigma_k k} = \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(L)}_{\sigma_{k-1}, r, k-1} = 1.$$

Constraint (5.8) is redundant with implied Constraints (C.3) and (C.5):

$$\forall\ k \in \{1, \ldots, \overline{N}\},\ \sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k, r, k} = 1.$$

Constraint (5.9a) is redundant with implied Constraints (C.3), (C.5) and (C.4):

$$\forall\ s \in \mathcal{S}^{(L)}, \begin{cases} \text{if } s = \sigma_1,\ \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_1 r 1} - d^{i}_{\sigma_1} = \sum_{r \in \overline{E}_{\nu_1}} d^{(L)}_{\sigma_1 r 1} - 1 = 1 - 1 = 0. \\ \text{if } s \neq \sigma_1,\ \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{sr1} - d^{i}_{s} = 0 - 0 = 0. \end{cases}$$

Constraint (5.9b) is redundant with implied Constraints (C.3), (C.6), (C.8) and (C.4):

$$\begin{matrix} \forall\ s \in \mathcal{S}^{(L)} \\ \forall\ k \in \{2, \ldots, \overline{N}\} \end{matrix}, \begin{cases} \text{if } s = \sigma_k,\ \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_k r k} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{r \sigma_k k} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} - \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(L)}_{\sigma_{k-1}, r, k-1} = 1 - 1 = 0. \\ \text{if } s \neq \sigma_k,\ \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{srk} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{rsk} = 0 - 0 = 0. \end{cases}$$

Constraint (5.10) is redundant with implied Constraints (C.3), (C.6), (C.8) and (C.4):

$$\begin{matrix} \forall\ r \in \mathcal{S}^{(E)} \\ \forall\ k \in \{2, \ldots, \overline{N}\} \end{matrix}, \begin{cases} \text{if } r \in \overline{E}_{\nu_k},\ \displaystyle\sum_{s \in \mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = d^{(L)}_{\sigma_{k-1}, r, k-1} - d^{(L)}_{\sigma_{k-1}, r, k-1} = 0. \\ \text{if } r \notin \overline{E}_{\nu_k},\ \displaystyle\sum_{s \in \mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = d^{(E)}_{r \sigma_k k} - d^{(L)}_{\sigma_{k-1}, r, k-1} = 0 - 0 = 0. \end{cases}$$

Constraint (5.11) is redundant with implied Constraints (C.10) and (C.11):

$$\forall\ r \in \mathcal{S}_B, \begin{cases} \text{if } r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z_r}, \ldots, Z\}} f_{rz} = 1, \\ \text{if } r \in\!\!\notin \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z_r}, \ldots, Z\}} f_{rz} = f_{r \widetilde{z_r}} = 1. \end{cases}$$

Constraint (5.12) is redundant with implied Constraints (C.3), (C.9) and (C.11):

$$\forall\ r \in \mathcal{S}_B,\ \begin{cases} \text{if } r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1,\ldots,\overline{N}\}}} d^{(L)}_{srk} = \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k rk} = \widetilde{z}_r. \\ \text{if } r \notin \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1,\ldots,\overline{N}\}}} d^{(L)}_{srk} = \widetilde{z}_r - 0 = \widetilde{z}_r. \end{cases}$$

Constraint (5.13) is redundant with implied Constraints (C.3) and (C.4):

$$\begin{matrix} \forall\ n \in \{1, \ldots, \overline{N}\} \\ \forall\ s \in L_n \\ \forall\ k \in \{1, \ldots, \overline{N}\} \end{matrix},\ \begin{cases} \text{if } n = \nu_k \text{ and } s = \sigma_k,\ \displaystyle\sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k rk} - w_{\nu_k \sigma_k k} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k rk} - 1 = 0 \geqslant 0. \\ \text{if } n \neq \nu_k \text{ or } s \neq \sigma_k,\ \displaystyle\sum_{r \in E_n} d^{(L)}_{srk} - w_{nsk} = \sum_{r \in E_n} d^{(L)}_{srk} \geqslant 0. \end{cases}$$

Recall that by definition if $r \in \mathcal{S}_R \cap \overline{E}_{\nu_k}$ then $\exists\ k' \in \{1, \ldots, k-1\}$ such that $\nu_{k'} = m_r$, i.e., $w_{m_r rk'} = 1$ so $\displaystyle\sum_{k' \in \{1,\ldots,k-1\}} w_{m_r rk'} = 1$. Thus, Constraint (5.14) is redundant with implied Constraints (C.3):

$$\begin{matrix} \forall\ r \in \mathcal{S}_R \\ \forall\ k \in \{1, \ldots, \overline{N}\} \end{matrix},\ \begin{cases} \text{if } r \in \overline{E}_{\nu_k},\ \displaystyle\sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k' \in \{1,\ldots,k-1\}} w_{m_r rk'} \leqslant d^{(L)}_{\sigma_k rk} - 1 \leqslant 0. \\ \text{if } r \notin \overline{E}_{\nu_k},\ \displaystyle\sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k' \in \{1,\ldots,k-1\}} w_{m_r rk'} = 0 - \sum_{k' \in \{1,\ldots,k-1\}} w_{m_r rk'} \leqslant 0. \end{cases}$$

Thus $\Pi(w)$ is equivalent to

$$\min_{D \in \{0,1\}} \left(c^T D\right)$$

$$s.t.\ D \text{ satisfice Constraints (C.3)-(C.11)}$$

Equivalence of $\overline{\Pi}(w)$ and $\Pi(w)$. Among Constraints (C.3)-(C.11), some of them fix variables to 0 or 1. In summary, we have

$$\forall\ k \in \{1, \ldots, \overline{N}\},\ \begin{cases} \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0, \\ \qquad \forall r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k rk} = 0. \end{cases}$$

$$\begin{cases} d^i_{\sigma_1} = 1, \\ \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\}, d^i_s = 0. \end{cases}$$

$$\forall\ k \in \{2, \ldots, \overline{N}\},\ \begin{cases} \forall\ r \in \mathcal{S}^{(E)},\ \forall s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ d^{(E)}_{rsk} = 0, \\ \forall\ r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} = 0, \end{cases}$$

$$\forall r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B,\ \begin{cases} f_{r\widetilde{z}_r} = 1, \\ \forall\ z \in \{\widetilde{z}_r + 1, \ldots, Z\}, f_{rz} = 0. \end{cases}$$

These constraints and their associated variables can be treated as constant for the sub-problem $\Pi(w)$. By deleting these constraints and variables, we get that $\Pi(w)$ is equivalent to the optimization problem $\Pi_1(w)$ defined as

$$\begin{cases} \min \left(\displaystyle\sum_{\substack{k \in \{2,\ldots,\overline{N}\} \\ r \in \overline{E}_{\nu_{k-1}}}} t^{(E)}_{r\sigma_k} d^{(E)}_{r\sigma_k k} + \sum_{\substack{k \in \{1,\ldots,\overline{N}\} \\ r \in \overline{E}_{\nu_k}}} t^{(L)}_{\sigma_k r} d^{(L)}_{\sigma_k r k} + \sum_{\substack{r \in \overline{\mathcal{S}}_B \\ z \in \{\widetilde{z}_r, \ldots, Z\}}} \beta_z f_{rz} \right) \\ s.t. \begin{cases} \forall\ k \in \{1, \ldots, \overline{N}\},\ \displaystyle\sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} = 1, \\ \forall\ k \in \{2, \ldots, \overline{N}\},\ \forall\ r \in \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} - d^{(L)}_{\sigma_{k-1}, r, k-1} = 0, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k} = \widetilde{z}_r, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1. \end{cases} \end{cases}$$

where the cost is the equal to $c^T D$ minus the constant $t^{(E)}_{s^i \sigma_1} + \displaystyle\sum_{r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B} \alpha_{\widetilde{z}_r}$. Using the second constraint of $\Pi_1(w)$ (which also is implied Constraint (C.7)), we can replace

$d^{(E)}_{r\sigma_k k}$ by $d^{(L)}_{\sigma_{k-1},r,k-1}$, thus $\Pi_1(w)$ is equivalent to the problem $\Pi_2(w)$ defined as

$$\left\{ \min \left(\sum_{\substack{k\in\{1,\dots,\overline{N}-1\} \\ r\in\overline{E}_{\nu_k}}} \left(t^{(E)}_{r\sigma_{k+1}} + t^{(L)}_{\sigma_k r} \right) d^{(L)}_{\sigma_k rk} + \sum_{\substack{k=\overline{N} \\ r\in\overline{E}_{\nu_k}}} t^{(L)}_{\sigma_k r} d^{(L)}_{\sigma_k rk} + \sum_{\substack{r\in\overline{\mathcal{S}}_B \\ z\in\{\widetilde{z}_r,\dots,Z\}}} \beta_z f_{rz} \right) \right.$$

$$s.t. \begin{cases} \forall\ k \in \{1,\dots,\overline{N}\},\ \displaystyle\sum_{r\in\overline{E}_{\nu_k}} d^{(L)}_{\sigma_k rk} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f_{rz} - \sum_{k\in\overline{K}_r} d^{(L)}_{\sigma_k rk} = \widetilde{z}_r, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f_{rz} = 1. \end{cases}$$

By changing the variables of $\Pi_2(w)$ such that

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \forall\ r \in \overline{E}_{\nu_k},\ \overline{d}_{rk} = d^{(L)}_{\sigma_k rk},$$

and

$$\forall\ s \in \overline{\mathcal{S}}_B,\ \forall\ z \in \{\widetilde{z}_r,\dots,Z\},\ \overline{f}_{rz} = f_{rz},$$

and by definition of $\overline{t}_{rk}$, we obtain $\overline{\Pi}(w)$. Consequently, $\overline{\Pi}(w)$ and $\Pi_2(w)$ are identical. Since $\Pi(w)$ is equivalent to $\Pi_1(w)$, itself equivalent to $\Pi_2(w)$, we have proven that $\Pi(w)$ and $\overline{\Pi}(w)$ are equivalent problems.

□

Theorem 3. *Let $w \in \mathcal{W} \cap \{0,1\}$ and $D^* = (d^*, f^*)$ be an extreme point of $\overline{\mathcal{L}}(w)$, then*

$$d^* \in \{0,1\}.$$

Proof. Let $D^* = (d^*, f^*)$ be an extreme point of $\overline{\mathcal{L}}(w)$ (not necessarily optimal). Let us prove by contradiction that $d^* \in \{0,1\}$. We suppose by contradiction that there exists $l \in \{1, \dots, \overline{N}\}$ and $p \in \overline{E}_{\nu_l}$ such that $d^*_{pl} \notin \{0,1\}$. Since $D^* \in \overline{\mathcal{L}}(w)$, we have

$$1 = \sum_{r \in \overline{E}_{\nu_l}} d^*_{rl} = d^*_{pl} + \sum_{r \in \overline{E}_{\nu_l} \setminus \{p\}} d^*_{rl}$$

Therefore, we have $\sum_{r \in \overline{E}_{\nu_l} \setminus \{p\}} d^*_{rl} \notin \{0,1\}$. Since $0 \leqslant d^*_{rl} \leqslant 1$, there exists $q \in \overline{E}_{\nu_l} \setminus \{p\}$ such that $d^*_{ql} \notin \{0,1\}$. We consider two distinct cases:

If $\nu_l \in \mathcal{N}_r$, then $E_{\nu_l} \cap \mathcal{S}_B = \emptyset$, thus $\overline{E}_{\nu_l} \cap \overline{\mathcal{S}}_B = \emptyset$, thus $p, q \notin \overline{\mathcal{S}}_B$. Consider

$$\epsilon = \min\left\{d^*_{pl},\ 1 - d^*_{pl},\ d^*_{ql},\ 1 - d^*_{ql}\right\} > 0,$$

and $D^1 = (d^1, f^*)$ and $D^2 = (d^2, f^*)$ such that

$$d^1_{pl} = d^*_{pl} - \epsilon \qquad d^2_{pl} = d^*_{pl} + \epsilon$$
$$d^1_{ql} = d^*_{ql} + \epsilon \qquad d^2_{ql} = d^*_{ql} - \epsilon$$

otherwise $d^1_{rk} = d^2_{rk} = d^*_{rk}$. Let us show that $D^1, D^2 \in \overline{\mathcal{L}}(w)$. First, by definition of ϵ, $0 \leqslant d^1, d^2 \leqslant 1$. Moreover, let us prove that $\forall k \in \{1, \dots, \overline{N}\}$, $\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}$. Indeed, let $k \in \{1, \dots, \overline{N}\}$

- If $k \neq l$, then $\forall\ r \in \overline{E}_{\nu_k}$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus $\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}$.

- If $k = l$, then we have

$$\sum_{r\in\overline{E}_{\nu_l}} d^1_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^1_{rl} + d^1_{pl} + d^1_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} - \epsilon + d^*_{ql} + \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl},$$

and

$$\sum_{r\in\overline{E}_{\nu_l}} d^2_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^2_{rl} + d^2_{pl} + d^2_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} + \epsilon + d^*_{ql} - \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl}.$$

Since $p, q \notin \overline{\mathcal{S}}_B$, then $\forall r \in \overline{\mathcal{S}}_B$, $\sum_{k\in\{1,\ldots,\overline{N}\}} d^1_{rk} = \sum_{k\in\{1,\ldots,\overline{N}\}} d^2_{rk} = \sum_{k\in\{1,\ldots,\overline{N}\}} d^*_{rk}$. Therefore, since only these sums involve the variable $\overline{d}$ in the constraints of $\overline{\mathcal{L}}(w)$, and that $D^* \in \overline{\mathcal{L}}(w)$, then $D^1, D^2 \in \overline{\mathcal{L}}(w)$. However, it is clear that $d^* = \frac{1}{2}(d^1 + d^2)$ so $D^* = \frac{1}{2}(D^1 + D^2)$, and since $\epsilon > 0$, $D^1 \neq D^2 \neq D^*$, In conclusion, D^* is not an extreme point which leads to a contradiction.

If $\nu_l \notin \mathcal{N}_r$, then $E_{\nu_l} \subset \mathcal{S}_B$, thus $\overline{E}_{\nu_l} \subset \overline{\mathcal{S}}_B$, so $p, q \in \overline{\mathcal{S}}_B$. The proof of Theorem 3 under this case requires a technical lemma (Lemma 14). Let us define

$$\forall\ k \in \{1,\ldots,\overline{N}\},\ \mathcal{R}_k = \left\{ r \in \overline{E}_{\nu_k} \mid d^*_{rk} \notin \{0,1\} \right\} \text{ and } \mathcal{R} = \bigcup_{k\in\{1,\ldots,\overline{N}\}} \mathcal{R}_k.$$

By definition we have $p, q \in \mathcal{R}_l$ so $|\mathcal{R}| > 0$. We also define

$$\forall\ r \in \mathcal{R},\ \mathcal{K}_r = \left\{ k \in \overline{K}_r \mid r \in \mathcal{R}_k \right\} \text{ and } \mathcal{K} = \bigcup_{r\in\mathcal{R}} \mathcal{K}_r.$$

such that $l \in \mathcal{K}_p \cap \mathcal{K}_q$ and $l \in \mathcal{K}$. By definition if $r \in \mathcal{R}_k$, then $k \in \mathcal{K}_r$.

Definition 2. *We say that $r \in \mathcal{R}$ is a stack linked through fractional solutions to stack p at stage l if $r = p$ or if there exists $J \in \{2,\ldots,|\mathcal{K}|\}$ and two sequences $(k_1,\ldots,k_J) \in \mathcal{K}$ and $(r_1,\ldots,r_J) \in \mathcal{R}$ such that:*

(i) $k_1 \neq \ldots \neq k_J$ and $r_1 \neq \ldots \neq r_J$.

(ii) $k_1 = l$ *and* $r_1 = p$.

(iii) $\forall\ j \in \{1, \ldots, J-1\}$, $r_j \in \mathcal{R}_{k_j} \cap \mathcal{R}_{k_{j+1}}$, *i.e.,* $d^*_{r_j k_j} \notin \{0,1\}$ *and* $d^*_{r_j k_{j+1}} \notin \{0,1\}$.

(iv) $r_J = r$ *and* $r \in \mathcal{R}_{k_J}$, *i.e.,* $d^*_{r k_J} \notin \{0,1\}$.

We denote by $\mathcal{T}_{p,l}$ *the set of stacks linked through fractional solutions to stack* p *at stage* l.

We now state the technical lemma to identify two sub-cases:

Lemma 14. *Let* $\mathcal{T}_{p,l}$ *the set of stacks linked through fractional solutions to stack* p *at stage* l*, then at least one of the two following statements is true:*

($*$*)* $q \in \mathcal{T}_{p,l}$.

($**$*)* *there exists* $r \in \mathcal{T}_{p,l}$ *such that* $\sum_{k \in \overline{K}_r} d^*_{rk} \notin \mathbb{N}$.

Before proving Lemma 14, we assume this lemma holds and concludes the proof of Theorem 3. Therefore, using Lemma 14, we consider two sub-cases:

If Condition (*) holds, since $p \neq q$, then we know there exists $J \in \{2, \ldots, |\mathcal{K}|\}$ and two sequences $(k_1, \ldots, k_J) \in \mathcal{K}$ and $(r_1, \ldots, r_J) \in \mathcal{R}$ such that: *(i)* $k_1 \neq \ldots \neq k_J$ and $r_1 \neq \ldots \neq r_J$, *(ii)* $k_1 = l$ and $r_1 = p$, *(iii)* $\forall\ j \in \{1, \ldots, J-1\}$, $r_j \in \mathcal{R}_{k_j} \cap \mathcal{R}_{k_{j+1}}$, and *(iv)* $r_J = q$ and $q \in \mathcal{R}_{k_J}$. Now consider

$$\epsilon = \min\left\{ \min_{j \in \{1,\ldots,J-1\}} \left\{ d^*_{r_j k_j},\ 1 - d^*_{r_j k_j},\ d^*_{r_j k_{j+1}},\ 1 - d^*_{r_j k_{j+1}} \right\},\ d^*_{q k_J},\ 1 - d^*_{q k_J},\ d^*_{ql},\ 1 - d^*_{ql} \right\} > 0,$$

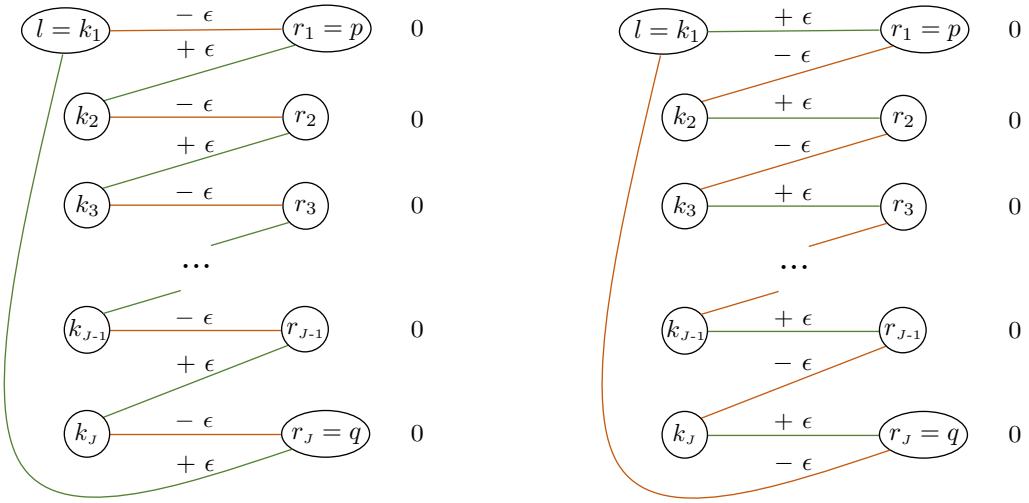
and $D^1 = (d^1, f^*)$ and $D^2 = (d^2, f^*)$ such that

$$\forall\ j \in \{1, \ldots, J-1\},\ \begin{cases} d^1_{r_j k_j} = d^*_{r_j k_j} - \epsilon \\ d^1_{r_j k_{j+1}} = d^*_{r_j k_{j+1}} + \epsilon \end{cases} \qquad \forall\ j \in \{1, \ldots, J-1\},\ \begin{cases} d^2_{r_j k_j} = d^*_{r_j k_j} + \epsilon \\ d^2_{r_j k_{j+1}} = d^*_{r_j k_{j+1}} - \epsilon \end{cases}$$

$$d^1_{q k_J} = d^*_{q k_J} - \epsilon \qquad d^2_{q k_J} = d^*_{q k_J} + \epsilon$$

$$d^1_{ql} = d^*_{ql} + \epsilon \qquad d^2_{ql} = d^*_{ql} - \epsilon$$

otherwise $d^1_{rk} = d^2_{rk} = d^*_{rk}$. We illustrate the changes made in D^1 and D^2 compared to D^* in Figure C-1. Let us show that $D^1, D^2 \in \overline{\mathcal{L}}(w)$. First, by definition of ϵ,



(a) Illustration of D^1.

(b) Illustration of D^2.

Figure C-1: Illustration of two feasible points D^1 and D^2 such that their average is an extreme point D^* in the case where Condition $(*)$ holds. Numbers on the right show the change of balance for nodes $(r_j)_{j\in\{1,\dots,J\}}$.

$0 \leqslant d^1, d^2 \leqslant 1$. Moreover, let us prove that $\forall k \in \{1, \dots, \overline{N}\}$, $\sum_{r\in\overline{E}_{\nu_k}} d^1_{rk} = \sum_{r\in\overline{E}_{\nu_k}} d^2_{rk} = \sum_{r\in\overline{E}_{\nu_k}} d^*_{rk}$. Indeed, let $k \in \{1, \dots, \overline{N}\}$

- If $\forall j \in \{2, \dots, J\}$, $k \neq k_j$ and $k \neq l$, then $\forall\, r \in \overline{E}_{\nu_k}$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus

$$\sum_{r\in\overline{E}_{\nu_k}} d^1_{rk} = \sum_{r\in\overline{E}_{\nu_k}} d^2_{rk} = \sum_{r\in\overline{E}_{\nu_k}} d^*_{rk}.$$

- If $\exists j \in \{2, \dots, J\}$, $k = k_j$, then we have

$$\sum_{r\in\overline{E}_{\nu_{k_j}}} d^1_{rk_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}} \\ r\neq r_j, r_{j-1}}} d^1_{rk_j} + d^1_{r_jk_j} + d^1_{r_{j-1}k_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}} \\ r\neq r_j, r_{j-1}}} d^*_{rk_j} + d^*_{r_jk_j} - \epsilon + d^*_{r_{j-1}k_j} + \epsilon = \sum_{r\in\overline{E}_{\nu_{k_j}}} d^*_{rk_j},$$

and

$$\sum_{r\in\overline{E}_{\nu_{k_j}}} d^2_{rk_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^2_{rk_j} + d^2_{r_jk_j} + d^2_{r_{j-1}k_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^*_{rk_j} + d^*_{r_jk_j} + \epsilon + d^*_{r_{j-1}k_j} - \epsilon = \sum_{r\in\overline{E}_{\nu_{k_j}}} d^*_{rk_j}.$$

- If $k = l$, then we have

$$\sum_{r\in\overline{E}_{\nu_l}} d^1_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^1_{rl} + d^1_{pl} + d^1_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} - \epsilon + d^*_{ql} + \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl},$$

and

$$\sum_{r\in\overline{E}_{\nu_l}} d^2_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^2_{rl} + d^2_{pl} + d^2_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} + \epsilon + d^*_{ql} - \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl}.$$

Moreover, let us now show that $\forall\ r \in \overline{\mathcal{S}}_B$, $\sum_{k\in\overline{K}_r} d^1_{rk} = \sum_{k\in\overline{K}_r} d^2_{rk} = \sum_{k\in\overline{K}_r} d^*_{rk}$. Indeed, let $r \in \overline{\mathcal{S}}_B$,

- If $\forall j \in \{1, \ldots, J-1\}$, $r \neq r_j$ and $r \neq q$, then we have $\forall\ k \in \overline{K}_r$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus

$$\sum_{k\in\overline{K}_r} d^1_{rk} = \sum_{k\in\overline{K}_r} d^2_{rk} = \sum_{k\in\overline{K}_r} d^*_{rk}.$$

- If $\exists j \in \{1, \ldots, J-1\}$, $r = r_j$, then we have

$$\sum_{k\in\overline{K}_{r_j}} d^1_{r_jk} = \sum_{\substack{k\in\overline{K}_{r_j}\\ k\neq k_j, k_{j+1}}} d^1_{r_jk} + d^1_{r_jk_j} + d^1_{r_jk_{j+1}} = \sum_{\substack{k\in\overline{K}_{r_j}\\ k\neq k_j, k_{j+1}}} d^*_{r_jk} + d^*_{r_jk_j} - \epsilon + d^*_{r_jk_{j+1}} + \epsilon = \sum_{k\in\overline{K}_{r_j}} d^*_{r_jk},$$

and

$$\sum_{k\in\overline{K}_{r_j}} d^2_{r_jk} = \sum_{\substack{k\in\overline{K}_{r_j}\\ k\neq k_j, k_{j+1}}} d^2_{r_jk} + d^2_{r_jk_j} + d^2_{r_jk_{j+1}} = \sum_{\substack{k\in\overline{K}_{r_j}\\ k\neq k_j, k_{j+1}}} d^*_{r_jk} + d^*_{r_jk_j} + \epsilon + d^*_{r_jk_{j+1}} - \epsilon = \sum_{k\in\overline{K}_{r_j}} d^*_{r_jk}.$$

- If $r = q$, then we have

$$\sum_{k\in\overline{K}_q} d^1_{qk} = \sum_{\substack{k\in\overline{K}_q\\ k\neq k_J,l}} d^1_{qk} + d^1_{qk_J} + d^1_{ql} = \sum_{\substack{k\in\overline{K}_q\\ k\neq k_J,l}} d^*_{qk} + d^*_{qk_J} - \epsilon + d^*_{ql} + \epsilon = \sum_{k\in\overline{K}_q} d^*_{qk},$$

and

$$\sum_{k\in\overline{K}_q} d^2_{qk} = \sum_{\substack{k\in\overline{K}_q\\ k\neq k_J,l}} d^2_{qk} + d^2_{qk_J} + d^2_{ql} = \sum_{\substack{k\in\overline{K}_q\\ k\neq k_J,l}} d^*_{qk} + d^*_{qk_J} + \epsilon + d^*_{ql} - \epsilon = \sum_{k\in\overline{K}_q} d^*_{qk},$$

Therefore, since only these sums involve the variable $\overline{d}$ in the constraints of $\overline{\mathcal{L}}(w)$, and that $D^* \in \overline{\mathcal{L}}(w)$, then $D^1, D^2 \in \overline{\mathcal{L}}(w)$. However, it is clear that $d^* = \frac{1}{2}(d^1 + d^2)$ so $D^* = \frac{1}{2}(D^1 + D^2)$, and since $\epsilon > 0$, $D^1 \neq D^2 \neq D^*$, In conclusion, D^* is not an extreme point which leads to a contradiction.

If Condition $(*)$ does not hold but Condition $(**)$ does, then there exists $s \in \mathcal{T}_{p,l}$ such that $\sum_{k\in\overline{K}_s} d^*_{sk} \notin \mathbb{N}$. Since $(d^*, f^*) \in \mathcal{L}(w)$, we have

$$\sum_{z\in\{\widetilde{z}_s,\ldots,Z\}} z f^*_{sz} = \widetilde{z}_s + \sum_{k\in\overline{K}_s} d^*_{sk} \notin \mathbb{N}.$$

This implies that, if we consider

$$\overline{z_s} = \max\left\{z \in \{\widetilde{z}_s,\ldots,Z\} \mid f^*_{sz} > 0\right\} \text{ and } \underline{z_s} = \min\left\{z \in \{\widetilde{z}_s,\ldots,Z\} \mid f^*_{sz} > 0\right\},$$

then we have

$$\overline{z_s} > \underline{z_s},$$

and consequently

$$f^*_{s\overline{z_s}} \notin \{0,1\} \text{ and } f^*_{s\underline{z_s}} \notin \{0,1\}.$$

Moreover, since $s \in \mathcal{T}_{p,l}$, then there exists $J \in \{2,\ldots,|\mathcal{K}|\}$ and two sequences $(k_1,\ldots,k_J) \in \mathcal{K}$ and $(s_1,\ldots,s_J) \in \mathcal{R}$ such that: *(i)* $k_1 \neq \ldots \neq k_J$ and $s_1 \neq \ldots \neq s_J$,

(ii) $k_1 = l$ and $s_1 = p$, *(iii)* $\forall\ j \in \{1, \ldots, J-1\}$, $s_j \in \mathcal{R}_{k_j} \cap \mathcal{R}_{k_{j+1}}$, and *(iv)* $s_J = s$ and $s \in \mathcal{R}_{k_J}$.

Similarly, we can use Lemma 14 with q. Since Condition $(*)$ does not hold, then $q \notin \mathcal{T}_{p,l}$, and thus $p \neq \mathcal{T}_{q,l}$. Therefore, we know that there exists $t \in \mathcal{T}_{q,l}$ such that $\sum_{k \in \overline{K}_t} d^*_{tk} \notin \mathbb{N}$. With the same argument, we know that if

$$\overline{z_t} = \max\left\{z \in \{\widetilde{z}_t, \ldots, Z\} \mid f^*_{tz} > 0\right\} \text{ and } \underline{z_t} = \min\left\{z \in \{\widetilde{z}_t, \ldots, Z\} \mid f^*_{tz} > 0\right\},$$

then

$$\overline{z_t} > \underline{z_t},\ f^*_{t\overline{z_t}} \notin \{0,1\} \text{ and } f^*_{t\underline{z_t}} \notin \{0,1\}.$$

In addition, there exists $I \in \{2, \ldots, |\mathcal{K}|\}$ and two sequences $(l_1, \ldots, l_I) \in \mathcal{K}$ and $(t_1, \ldots, t_I) \in \mathcal{R}$ such that: *(i)* $l_1 \neq \ldots \neq l_I$ and $t_1 \neq \ldots \neq t_I$, *(ii)* $l_1 = l$ and $t_1 = q$, *(iii)* $\forall\ i \in \{1, \ldots, I-1\}$, $t_i \in \mathcal{R}_{l_i} \cap \mathcal{R}_{l_{i+1}}$, and *(iv)* $t_I = t$ and $t \in \mathcal{R}_{l_I}$.

Note that $q \notin \mathcal{T}_{p,l}$ implies that $\forall (j,i) \in \{1, \ldots, J\} \times \{1, \ldots, I\}$, $s_j \neq t_i$ and $k_j \neq l_i$. We now consider

$$\epsilon = \min \left\{ \begin{array}{c} \min\limits_{j \in \{1,\ldots,J-1\}} \left\{d^*_{s_j k_j},\ 1 - d^*_{s_j k_j},\ d^*_{s_j k_{j+1}},\ 1 - d^*_{s_j k_{j+1}}\right\},\ d^*_{s_J k_J},\ 1 - d^*_{s_J k_J}, \\ \min\limits_{i \in \{1,\ldots,I-1\}} \left\{d^*_{t_i l_i},\ 1 - d^*_{t_i l_i},\ d^*_{t_i l_{i+1}},\ 1 - d^*_{t_i l_{i+1}}\right\},\ d^*_{t_I l_I},\ 1 - d^*_{t_I l_I}, \\ \left(\overline{z_s} - \underline{z_s}\right) f^*_{s\underline{z_s}},\ \left(\overline{z_s} - \underline{z_s}\right)\left(1 - f^*_{s\underline{z_s}}\right),\ \left(\overline{z_s} - \underline{z_s}\right) f^*_{s\overline{z_s}},\ \left(\overline{z_s} - \underline{z_s}\right)\left(1 - f^*_{s\overline{z_s}}\right), \\ \left(\overline{z_t} - \underline{z_t}\right) f^*_{t\underline{z_t}},\ \left(\overline{z_t} - \underline{z_t}\right)\left(1 - f^*_{t\underline{z_t}}\right),\ \left(\overline{z_t} - \underline{z_t}\right) f^*_{t\overline{z_t}},\ \left(\overline{z_t} - \underline{z_t}\right)\left(1 - f^*_{t\overline{z_t}}\right), \end{array} \right\} > 0.$$

and $D^1 = (d^1, f^1)$ and $D^2 = (d^2, f^2)$ such that

$$\forall\, j \in \{1, \dots, J-1\},\ \begin{cases} d^1_{s_j k_j} = d^*_{s_j k_j} - \epsilon \\ d^1_{s_j k_{j+1}} = d^*_{s_j k_{j+1}} + \epsilon \end{cases} \qquad \forall\, j \in \{1, \dots, J-1\},\ \begin{cases} d^2_{s_j k_j} = d^*_{s_j k_j} + \epsilon \\ d^2_{s_j k_{j+1}} = d^*_{s_j k_{j+1}} - \epsilon \end{cases}$$

$$\forall\, i \in \{1, \dots, I-1\},\ \begin{cases} d^1_{t_i l_i} = d^*_{t_i l_i} + \epsilon \\ d^1_{t_i l_{i+1}} = d^*_{t_i l_{i+1}} - \epsilon \end{cases} \qquad \forall\, i \in \{1, \dots, I-1\},\ \begin{cases} d^2_{t_i l_i} = d^*_{t_i l_i} - \epsilon \\ d^2_{t_i l_{i+1}} = d^*_{t_i l_{i+1}} + \epsilon \end{cases}$$

$$d^1_{s k_J} = d^*_{s k_J} - \epsilon \qquad d^2_{s k_J} = d^*_{s k_J} + \epsilon$$

$$d^1_{t l_I} = d^*_{t l_I} + \epsilon \qquad d^2_{t l_I} = d^*_{t l_I} - \epsilon$$

$$f^1_{s\underline{z_s}} = f^*_{s\underline{z_s}} + \frac{\epsilon}{\overline{z_s} - \underline{z_s}} \qquad f^2_{s\underline{z_s}} = f^*_{s\underline{z_s}} - \frac{\epsilon}{\overline{z_s} - \underline{z_s}}$$

$$f^1_{s\overline{z_s}} = f^*_{s\overline{z_s}} - \frac{\epsilon}{\overline{z_s} - \underline{z_s}} \qquad f^2_{s\overline{z_s}} = f^*_{s\overline{z_s}} + \frac{\epsilon}{\overline{z_s} - \underline{z_s}}$$

$$f^1_{t\underline{z_t}} = f^*_{t\underline{z_t}} - \frac{\epsilon}{\overline{z_t} - \underline{z_t}} \qquad f^2_{t\underline{z_t}} = f^*_{t\underline{z_t}} + \frac{\epsilon}{\overline{z_t} - \underline{z_t}}$$

$$f^1_{t\overline{z_t}} = f^*_{t\overline{z_t}} + \frac{\epsilon}{\overline{z_t} - \underline{z_t}} \qquad f^2_{t\overline{z_t}} = f^*_{t\overline{z_t}} - \frac{\epsilon}{\overline{z_t} - \underline{z_t}}$$

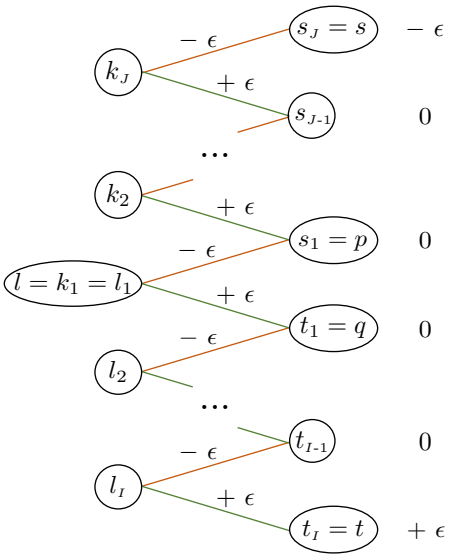
otherwise $d^1_{rk} = d^2_{rk} = d^*_{rk}$ and $f^1_{rz} = f^2_{rz} = f^*_{rz}$. We illustrate the changes made in D^1 and D^2 compared to D^* in Figure C-2.

Let us show that $D^1, D^2 \in \overline{\mathcal{L}}(w)$. First, by definition of ϵ, $0 \leqslant d^1, d^2 \leqslant 1$ and $0 \leqslant f^1, f^2 \leqslant 1$. Moreover, let us prove that $\forall k \in \{1, \dots, \overline{N}\}$, $\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}$. Indeed, let $k \in \{1, \dots, \overline{N}\}$

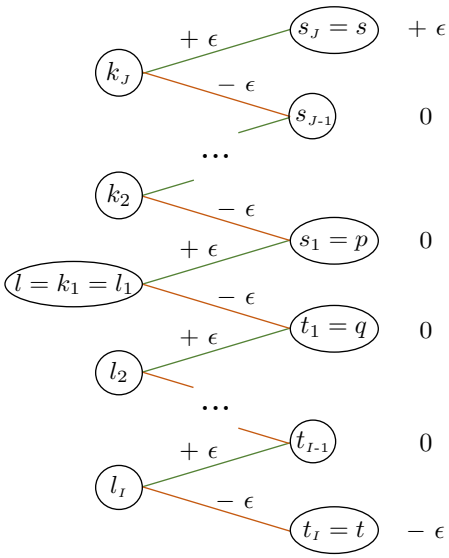
- If $\forall j \in \{2, \dots, J\}$, $k \neq k_j$ and $\forall i \in \{2, \dots, I\}$, $k \neq l_i$ and $k \neq l$, then $\forall\, r \in \overline{E}_{\nu_k}$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus

$$\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}.$$

- If $\exists j \in \{2, \dots, J\}$, $k = k_j$, then we have



(a) Illustration of D^1.



(b) Illustration of D^2.

Figure C-2: Illustration of two feasible points D^1 and D^2 such that their average is an extreme point D^* in the case where Condition $(*)$ does not hold but Condition $(**)$ does. Numbers on the right show the change of balance for nodes $(s_j)_{j\in\{1,\dots,J\}}$ and $(t_i)_{i\in\{1,\dots,I\}}$.

$$\sum_{r\in\overline{E}_{\nu_{k_j}}} d^1_{rk_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^1_{rk_j}+d^1_{r_jk_j}+d^1_{r_{j-1}k_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^*_{rk_j}+d^*_{r_jk_j}-\epsilon+d^*_{r_{j-1}k_j}+\epsilon = \sum_{r\in\overline{E}_{\nu_{k_j}}} d^*_{rk_j},$$

and

$$\sum_{r\in\overline{E}_{\nu_{k_j}}} d^2_{rk_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^2_{rk_j}+d^2_{r_jk_j}+d^2_{r_{j-1}k_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^*_{rk_j}+d^*_{r_jk_j}+\epsilon+d^*_{r_{j-1}k_j}-\epsilon = \sum_{r\in\overline{E}_{\nu_{k_j}}} d^*_{rk_j}.$$

- If $\exists i \in \{2,\dots,I\}$, $k = l_i$, then we have

$$\sum_{r\in\overline{E}_{\nu_{l_i}}} d^1_{rl_i} = \sum_{\substack{r\in\overline{E}_{\nu_{l_i}}\\ r\neq t_i, t_{i-1}}} d^1_{rl_i}+d^1_{t_il_i}+d^1_{t_{i-1}l_i} = \sum_{\substack{r\in\overline{E}_{\nu_{l_i}}\\ r\neq t_i, t_{i-1}}} d^*_{rl_i}+d^*_{t_il_i}+\epsilon+d^*_{t_{i-1}l_i}-\epsilon = \sum_{r\in\overline{E}_{\nu_{l_i}}} d^*_{rl_i},$$

and

$$\sum_{r\in\overline{E}_{\nu_{l_i}}} d^2_{rl_i} = \sum_{\substack{r\in\overline{E}_{\nu_{l_i}}\\ r\neq t_i,t_{i-1}}} d^2_{rl_i}+d^2_{t_il_i}+d^2_{t_{i-1}l_i} = \sum_{\substack{r\in\overline{E}_{\nu_{l_i}}\\ r\neq t_i,t_{i-1}}} d^*_{rl_i}+d^*_{t_il_i}-\epsilon+d^*_{t_{i-1}l_i}+\epsilon = \sum_{r\in\overline{E}_{\nu_{l_i}}} d^*_{rl_i},$$

- If $k = l = k_1 = l_1$, then we have

$$\sum_{r\in\overline{E}_{\nu_l}} d^1_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^1_{rl} + d^1_{pl} + d^1_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} - \epsilon + d^*_{ql} + \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl},$$

and

$$\sum_{r\in\overline{E}_{\nu_l}} d^2_{rl} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^2_{rl} + d^2_{pl} + d^2_{ql} = \sum_{\substack{r\in\overline{E}_{\nu_l}\\ r\neq p,q}} d^*_{rl} + d^*_{pl} + \epsilon + d^*_{ql} - \epsilon = \sum_{r\in\overline{E}_{\nu_l}} d^*_{rl}.$$

Moreover, let $r \in \overline{\mathcal{S}}_B$, we check that (d^1, f^1) and (d^2, f^2) verify the two types of constraints of $\overline{\mathcal{L}}(w)$ associated with r:

- If $r \neq s, t$, then $\forall z \in \{\widetilde{z}_r, \ldots, Z\}$, $f^1_{rz} = f^2_{rz} = f^*_{rz}$. In addition, we prove that

$$\sum_{k\in\overline{K}_r} d^1_{rk} = \sum_{k\in\overline{K}_r} d^2_{rk} = \sum_{k\in\overline{K}_r} d^*_{rk},$$

which by feasibility of D^* would prove that the two constraints associated with r are satisfied.

  - ◇ If $\forall j \in \{1, \ldots, J-1\}$, $r \neq r_j$ and $\forall i \in \{1, \ldots, I-1\}$, $r \neq t_i$, then $\forall\, k \in \overline{K}_r$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus

$$\sum_{k\in\overline{K}_r} d^1_{rk} = \sum_{k\in\overline{K}_r} d^2_{rk} = \sum_{k\in\overline{K}_r} d^*_{rk}.$$

  - ◇ If $\exists j \in \{1, \ldots, J-1\}$, $r = s_j$, then we have

$$\sum_{k\in\overline{K}_{s_j}} d^1_{s_jk} = \sum_{\substack{k\in\overline{K}_{s_j}\\ k\neq k_j,k_{j+1}}} d^1_{s_jk}+d^1_{s_jk_j}+d^1_{s_jk_{j+1}} = \sum_{\substack{k\in\overline{K}_{s_j}\\ k\neq k_j,k_{j+1}}} d^*_{s_jk}+d^*_{s_jk_j}-\epsilon+d^*_{s_jk_{j+1}}+\epsilon = \sum_{k\in\overline{K}_{s_j}} d^*_{s_jk},$$

and

$$\sum_{k\in\overline{K}_{s_j}} d^2_{s_jk} = \sum_{\substack{k\in\overline{K}_{s_j}\\ k\neq k_j,k_{j+1}}} d^2_{s_jk}+d^2_{s_jk_j}+d^2_{s_jk_{j+1}} = \sum_{\substack{k\in\overline{K}_{s_j}\\ k\neq k_j,k_{j+1}}} d^*_{s_jk}+d^*_{s_jk_j}+\epsilon+d^*_{s_jk_{j+1}}-\epsilon = \sum_{k\in\overline{K}_{s_j}} d^*_{s_jk}.$$

◇ If $\exists i \in \{1,\dots,I-1\}$, $r = t_i$, then we have

$$\sum_{k\in\overline{K}_{t_i}} d^1_{t_ik} = \sum_{\substack{k\in\overline{K}_{t_i}\\ k\neq l_i,l_{i+1}}} d^1_{t_ik} + d^1_{t_il_i} + d^1_{t_il_{i+1}} = \sum_{\substack{k\in\overline{K}_{t_i}\\ k\neq l_i,l_{i+1}}} d^*_{t_ik} + d^*_{t_il_i} + \epsilon + d^*_{t_il_{i+1}} - \epsilon = \sum_{k\in\overline{K}_{t_i}} d^*_{t_ik},$$

and

$$\sum_{k\in\overline{K}_{t_i}} d^2_{t_ik} = \sum_{\substack{k\in\overline{K}_{t_i}\\ k\neq l_i,l_{i+1}}} d^2_{t_ik} + d^2_{t_il_i} + d^2_{t_il_{i+1}} = \sum_{\substack{k\in\overline{K}_{t_i}\\ k\neq l_i,l_{i+1}}} d^*_{t_ik} + d^*_{t_il_i} - \epsilon + d^*_{t_il_{i+1}} + \epsilon = \sum_{k\in\overline{K}_{t_i}} d^*_{t_ik}.$$

- If $r = s$ or $r = t$, (both cases are similar, we only treat the case of $r = s$) we have

$$\begin{aligned}\sum_{z\in\{\widetilde{z}_s,\dots,Z\}} f^1_{sz} &= \sum_{\substack{z\in\{\widetilde{z}_s,\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^1_{sz} + f^1_{s\underline{z_s}} + f^1_{s\overline{z_s}}\\ &= \sum_{\substack{z\in\{\widetilde{z}_s,\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^*_{sz} + f^*_{s\underline{z_s}} + \frac{\epsilon}{\overline{z_s}-\underline{z_s}} + f^*_{s\overline{z_s}} - \frac{\epsilon}{\overline{z_s}-\underline{z_s}}\\ &= \sum_{z\in\{\widetilde{z}_s,\dots,Z\}} f^*_{sz} = 1,\end{aligned}$$

and

$$\begin{aligned}\sum_{z\in\{\widetilde{z_s},\ldots,Z\}} f^2_{sz} &= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} f^2_{sz} + f^2_{s\underline{z_s}} + f^2_{s\overline{z_s}} \\ &= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} f^*_{sz} + f^*_{s\underline{z_s}} - \frac{\epsilon}{\overline{z_s}-\underline{z_s}} + f^*_{s\overline{z_s}} + \frac{\epsilon}{\overline{z_s}-\underline{z_s}} \\ &= \sum_{z\in\{\widetilde{z_s},\ldots,Z\}} f^*_{sz} = 1.\end{aligned}$$

Moreover,

$$\begin{aligned}&\sum_{z\in\{\widetilde{z_s},\ldots,Z\}} z f^1_{sz} - \sum_{k\in\overline{K}_s} d^1_{sk} \\ &\quad= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} z f^1_{sz} + \underline{z_s} f^1_{s\underline{z_s}} + \overline{z_s} f^1_{s\overline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^1_{sk} + d^1_{sk_J}\right) \\ &\quad= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} z f^*_{sz} + \underline{z_s} f^*_{s\underline{z_s}} + \frac{\epsilon \underline{z_s}}{\overline{z_s}-\underline{z_s}} + \overline{z_s} f^*_{s\overline{z_s}} - \frac{\epsilon\overline{z_s}}{\overline{z_s}-\underline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^*_{sk} + d^*_{sk_J} - \epsilon\right) \\ &\quad= \sum_{z\in\{\widetilde{z_s},\ldots,Z\}} z f^*_{sz} - \sum_{k\in\overline{K}_s} d^*_{sk} = \widetilde{z_s}.\end{aligned}$$

Similarly,

$$\begin{aligned}&\sum_{z\in\{\widetilde{z_s},\ldots,Z\}} z f^2_{sz} - \sum_{k\in\overline{K}_s} d^2_{sk} \\ &\quad= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} z f^2_{sz} + \underline{z_s} f^2_{s\underline{z_s}} + \overline{z_s} f^2_{s\overline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^2_{sk} + d^2_{sk_J}\right) \\ &\quad= \sum_{\substack{z\in\{\widetilde{z_s},\ldots,Z\}\\ z\neq \underline{z_s},\overline{z_s}}} z f^*_{sz} + \underline{z_s} f^*_{s\underline{z_s}} - \frac{\epsilon \underline{z_s}}{\overline{z_s}-\underline{z_s}} + \overline{z_s} f^*_{s\overline{z_s}} + \frac{\epsilon\overline{z_s}}{\overline{z_s}-\underline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^*_{sk} + d^*_{sk_J} + \epsilon\right) \\ &\quad= \sum_{z\in\{\widetilde{z_s},\ldots,Z\}} z f^*_{sz} - \sum_{k\in\overline{K}_s} d^*_{sk} = \widetilde{z_s}.\end{aligned}$$

In conclusion, $D^1, D^2 \in \overline{\mathcal{L}}(w)$. However, it is clear that $d^* = \frac{1}{2}(d^1 + d^2)$ and $f^* = \frac{1}{2}(f^1 + f^2)$ so $D^* = \frac{1}{2}(D^1 + D^2)$, and since $\epsilon > 0$, $D^1 \neq D^2 \neq D^*$, In conclusion, D^* is not an extreme point which leads to a contradiction and concludes the proof of Theorem 3. $\square$

Lemma 14. *Let $\mathcal{T}_{p,l}$ the set of stacks linked through fractional solutions to stack p at stage l, then at least one of the two following statements is true:*

(*) $q \in \mathcal{T}_{p,l}$.

(**) there exists $r \in \mathcal{T}_{p,l}$ *such that* $\sum_{k \in \overline{K}_r} d^*_{rk} \notin \mathbb{N}$.

Proof of Lemma 14. Suppose by contradiction that $q \notin \mathcal{T}_{p,l}$ and $\forall\ r \in \mathcal{T}_{p,l}$, we have

$$\sum_{k \in \overline{K}_r} d^*_{rk} \in \mathbb{N}.$$

Let $r \in \mathcal{T}_{p,l} \subset \mathcal{R}$, then by definition we have $\forall\ k \in \overline{K}_r \backslash \mathcal{K}_r$, $d^*_{rk} \in \{0, 1\}$. Consequently,

$$\sum_{k \in \mathcal{K}_r} d^*_{rk} \in \mathbb{N}.$$

By summing on all $r \in \mathcal{T}_{p,l}$, then

$$\text{if } \Theta = \sum_{r \in \mathcal{T}_{p,l}} \sum_{k \in \mathcal{K}_r} d^*_{rk}, \text{ then } \Theta \in \mathbb{N}.$$

Now consider

$$\mathcal{K}_{p,l} = \{k \in \mathcal{K} \,|\exists\ r \in \mathcal{T}_{p,l} \text{ s.t. } k \in \mathcal{K}_r\}.$$

Let $k \in \mathcal{K}_{p,l}$, then we have

$$\{r \in \mathcal{T}_{p,l} \,|k \in \mathcal{K}_r\} = \{r \in \mathcal{T}_{p,l} \,|r \in \mathcal{R}_k\} = \mathcal{R}_k \cap \mathcal{T}_{p,l},$$

thus

$$\Theta = \sum_{r \in \mathcal{T}_{p,l}} \sum_{k \in \mathcal{K}_r} d^*_{rk} = \sum_{k \in \mathcal{K}_{p,l}} \sum_{r \in \left\{r' \in \mathcal{T}_{p,l} | k \in \mathcal{K}_{r'}\right\}} d^*_{rk} = \sum_{k \in \mathcal{K}_{p,l}} \sum_{r \in \mathcal{R}_k \cap \mathcal{T}_{p,l}} d^*_{rk}.$$

Let $k \in \mathcal{K}_{p,l} \setminus \{l\}$. Let us show that

$$\mathcal{R}_k \cap \mathcal{T}_{p,l} = \mathcal{R}_k.$$

To do so, we prove that $\mathcal{R}_k \subset \mathcal{T}_{p,l}$, i.e., let $s \in \mathcal{R}_k$ then we prove that $s \in \mathcal{T}_{p,l}$. If $s = p$, then $s \in \mathcal{T}_{p,l}$. Now suppose that $s \neq p$. First, since $k \in \mathcal{K}_{p,l}$, then by definition there exists $r \in \mathcal{T}_{p,l}$ such that $k \in \mathcal{K}_r$, thus $r \in \mathcal{R}_k$. If $r = s$, then $s \in \mathcal{T}_{p,l}$. Otherwise, we have $r \neq s$. In addition, since $k \neq l$, we can assume that $r \neq p$. Indeed, as we have shown for the existence of q in Theorem 4, if $|\mathcal{R}_k| \neq 0$, then $|\mathcal{R}_k| \geqslant 2$. Consequently, we take $r \in \mathcal{T}_{p,l}$ such that $r \neq p$. So there exists $J \in \{2, \ldots, |\mathcal{K}|\}$ and two sequences $(k_1, \ldots, k_J) \in \mathcal{K}$ and $(r_1, \ldots, r_J) \in \mathcal{R}$ that verifies properties *(i)*, *(ii)*, *(iii)* and *(iv)* $r_J = r$ and $r \in \mathcal{R}_{k_J}$. We consider three sub-cases. In each cases, we construct $I \in \{2, \ldots, |\mathcal{K}|\}$ and two new sequences $(l_1, \ldots, l_I) \in \mathcal{K}$, $(s_1, \ldots, s_I) \in \mathcal{R}$ that verifies properties *(i)*, *(ii)*, *(iii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I}$, hence getting the result $s \in \mathcal{T}_{p,l}$.

- ◇ If there exists $j \in \{2, \ldots, J\}$ such that $r_j = s$, then consider $I = j \in \{2, \ldots, |\mathcal{K}|\}$ and

$$(l_1, \ldots, l_I) = (k_1, \ldots, k_j) \in \mathcal{K} \text{ and } (s_1, \ldots, s_I) = (r_1, \ldots, r_j) \in \mathcal{R}.$$

  Clearly, $(l_1, \ldots, l_I)$ and $(s_1, \ldots, s_I)$ verify properties *(i)*, *(ii)*, *(iii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I}$.

- ◇ If $r \notin \{r_2, \ldots, r_J\}$ but there exists $j \in \{2, \ldots, J\}$ such that $k_j = k$. In this case, consider $I = j \in \{2, \ldots, |\mathcal{K}|\}$ and

$$(l_1, \ldots, l_I) = (k_1, \ldots, k_j) \in \mathcal{K} \text{ and } (s_1, \ldots, s_{I-1}, s_I) = (r_1, \ldots, r_{j-1}, s) \in \mathcal{R}.$$

  Clearly, $(l_1, \ldots, l_I)$ and $(s_1, \ldots, s_I)$ verify properties *(i)*, *(ii)*, *(iii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I}$.

- ◇ If $r \notin \{r_2, \ldots, r_J\}$ and $k \notin \{k_2, \ldots, k_J\}$. Since $k \notin \{k_1, \ldots, k_J\}$, then we must have $J < |\mathcal{K}|$. Consider $I = J + 1 \in \{2, \ldots, |\mathcal{K}|\}$ and

$$(l_1, \ldots, l_I) = (k_1, \ldots, k_J, k) \in \mathcal{K} \text{ and } (s_1, \ldots, s_I) = (r_1, \ldots, r_J, r) \in \mathcal{R}.$$

Clearly, $(l_1, \ldots, l_I)$ and $(s_1, \ldots, s_I)$ verify properties *(i)*, *(ii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I} = \mathcal{R}_k$. Note that $\forall\, i \in \{1, \ldots, I-1\}$, property *(iii)* holds by definition of $(k_1, \ldots, k_J)$ and $(r_1, \ldots, r_J)$. In addition, we have $s_{I-1} = r_J \in \mathcal{R}_{k_J} = \mathcal{R}_{s_{I-1}}$ by definition, and $s_{I-1} = r_J = r \in \mathcal{R}_k = \mathcal{R}_{s_I}$ which proves property *(iii)*.

In any case, $s \in \mathcal{T}_{p,l}$, and thus $\mathcal{R}_k \subset \mathcal{T}_{p,l}$ and $\mathcal{R}_k \cap \mathcal{T}_{p,l} = \mathcal{R}_k$. Using this fact, we have

$$\begin{aligned}
\Theta &= \sum_{k \in \mathcal{K}_{p,l}} \sum_{r \in \mathcal{R}_k \cap \mathcal{T}_{p,l}} d^*_{rk} \\
&= \sum_{k \in \mathcal{K}_{p,l} \setminus \{l\}} \sum_{r \in \mathcal{R}_k \cap \mathcal{T}_{p,l}} d^*_{rk} + \sum_{r \in \mathcal{R}_l \cap \mathcal{T}_{p,l}} d^*_{rl} \\
&= \sum_{k \in \mathcal{K}_{p,l} \setminus \{l\}} \sum_{r \in \mathcal{R}_k} d^*_{rk} + \sum_{r \in \mathcal{R}_l \cap \mathcal{T}_{p,l}} d^*_{rl} \\
&= \Theta_1 + \Theta_2,
\end{aligned}$$

where $\Theta_1 = \sum_{k \in \mathcal{K}_{p,l} \setminus \{l\}} \sum_{r \in \mathcal{R}_k} d^*_{rk}$ and $\Theta_2 = \sum_{r \in \mathcal{R}_l \cap \mathcal{T}_{p,l}} d^*_{rl}$. We know that $\Theta \in \mathbb{N}$ and we now show that $\Theta_1 \in \mathbb{N}$ but $\Theta_2 \notin \mathbb{N}$, which leads to the contradiction.

First, by definition we have $\mathcal{K}_{p,l} \setminus \{l\} \subset \{1, \ldots, \overline{N}\}$ and $\sum_{r \in \overline{E}_{\nu_k}} d^*_{rk} = \sum_{r \in \mathcal{R}_k} d^*_{rk}$. By feasibility of d^*, we must have

$$\forall\, k \in \mathcal{K}_{p,l} \setminus \{l\},\ \sum_{r \in \mathcal{R}_k} d^*_{rk} = 1 \in \mathbb{N},$$

which by summing over all $k \in \mathcal{K}_{p,l} \setminus \{l\}$ gives

$$\sum_{k \in \mathcal{K}_{p,l} \setminus \{l\}} \sum_{r \in \mathcal{R}_k} d^*_{rk} \in \mathbb{N},$$

so

$$\Theta_1 \in \mathbb{N}.$$

Second, we have by definition $p \in \mathcal{T}_{p,l} \cap \mathcal{R}_l$, so by non-negativity of d^*, we have

$$\Theta_2 = \sum_{r \in \mathcal{R}_l \cap \mathcal{T}_{p,l}} d^*_{rl} \geqslant d^*_{pl} > 0.$$

Similarly, since $q \in \mathcal{R}_l \setminus \mathcal{T}_{p,l}$ we have

$$\sum_{r \in \mathcal{R}_l \setminus \mathcal{T}_{p,l}} d^*_{rl} \geqslant d^*_{ql} > 0,$$

Using the feasibility of d^* we have

$$1 = \sum_{r \in \mathcal{R}_l} d^*_{rl} = \sum_{r \in \mathcal{R}_l \setminus \mathcal{T}_{p,l}} d^*_{rl} + \sum_{r \in \mathcal{R}_l \cap \mathcal{T}_{p,l}} d^*_{rl} = \sum_{r \in \mathcal{R}_l \setminus \mathcal{T}_{p,l}} d^*_{rl} + \Theta_2 > \Theta_2.$$

Therefore $0 < \Theta_2 < 1$, so

$$\Theta_2 \notin \mathbb{N}.$$

In conclusion, we have

$$\Theta = \Theta_1 + \Theta_2,$$

with $\Theta \in \mathbb{N}$, $\Theta_1 \in \mathbb{N}$ but $\Theta_2 \notin \mathbb{N}$, leading to a contradiction and proving Lemma 14. □

Theorem 4. *Let $w \in \mathcal{W} \cap \{0,1\}$. If D^* is an extreme point of $\overline{\mathcal{L}}(w)$ such that $D^* = \underset{\overline{D} \in \overline{\mathcal{L}}(w)}{argmin} \left(\overline{c}^T \overline{D}\right)$ and γ verifies Condition (A), then*

$$D^* \in \{0,1\}.$$

Proof. We have proven in Theorem 3. that if $D^* = (d^*, f^*)$ is an extreme point of $\overline{\mathcal{L}}(w)$, then $d^* \in \{0,1\}$. Now we further assume that D^* is optimal and that γ verifies Condition (A) to show that $f^* \in \{0,1\}$ by contradiction. Let $r \in \overline{\mathcal{S}}_B$, since $\forall\ k \in \overline{K}_r$, $d^*_{rk} \in \{0, \ldots, 1\}$, then we have $\sum_{k \in \overline{K}_r} d^*_{rk} \in \mathbb{N}$. Therefore, we have

$$\text{if } z^* = \widetilde{z_r} + \sum_{k \in \overline{K}_r} d^*_{rk}, \text{ then } z^* \in \mathbb{N}.$$

Note that we omit the r index in z^* just for the sake of clarity. Recall that since D^*

is feasible, we have

$$\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f^*_{rz} = \widetilde{z}_r + \sum_{k\in\overline{K}_r} d^*_{rk} = z^*.$$

Let us show that $f^*_{rz^*} = 1$, and $f^*_{rz} = 0$ otherwise. First, if $z^* = \widetilde{z}_r$ or $z^* = Z$, then the only feasible solution is $f^*_{rz^*} = 1$, and $f^*_{rz} = 0$ otherwise.

Otherwise, we have $z^* \in \{\widetilde{z}_r + 1, \dots, Z-1\}$. Let us suppose by contradiction that $f^*_{rz^*} < 1$. Then, since $\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f^*_{rz} = z^*$, there exist $\underline{z}$ and $\overline{z}$ such that $\underline{z} < z^* < \overline{z}$ with $f^*_{r\underline{z}} > 0$ and $f^*_{r\overline{z}} > 0$. Let

$$\epsilon = \min\left\{(\overline{z} - z^*)\, f^*_{r\overline{z}},\ (z^* - \underline{z})\, f^*_{r\underline{z}}\right\} > 0,$$

and consider

$$f_{r\overline{z}} = f^*_{r\overline{z}} - \frac{\epsilon}{\overline{z} - z^*},\ f_{r\underline{z}} = f^*_{r\underline{z}} - \frac{\epsilon}{z^* - \underline{z}} \text{ and } f_{rz^*} = f^*_{rz^*} + \frac{\epsilon}{\overline{z} - z^*} + \frac{\epsilon}{z^* - \underline{z}}.$$

Let us show that $(d^*, f) \in \overline{\mathcal{L}}(w)$. First, we have

$$\begin{aligned}
\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f_{rz} &= \sum_{\substack{z\in\{\widetilde{z}_r,\dots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} f_{rz} + f_{r\overline{z}} + f_{r\underline{z}} + f_{rz^*} \\
&= \sum_{\substack{z\in\{\widetilde{z}_r,\dots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} f^*_{rz} + f^*_{r\overline{z}} - \frac{\epsilon}{\overline{z} - z^*} + f^*_{r\underline{z}} - \frac{\epsilon}{z^* - \underline{z}} + f^*_{rz^*} + \frac{\epsilon}{\overline{z} - z^*} + \frac{\epsilon}{z^* - \underline{z}} \\
&= \sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f^*_{rz} = 1.
\end{aligned}$$

Moreover, by definition of ϵ, we have $1 \geqslant f^*_{r\overline{z}} \geqslant f_{r\overline{z}} \geqslant 0$, $1 \geqslant f^*_{r\underline{z}} \geqslant f_{r\underline{z}} \geqslant 0$, $f_{rz^*} \geqslant f^*_{rz^*} \geqslant 0$ and since $\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f_{rz} = 1$, we have $f_{rz^*} \leqslant 1$. Thus $0 \leqslant f \leqslant 1$.

Finally,

$$\begin{aligned}
\sum_{z\in\{\widetilde{z_r},\ldots,Z\}} z f_{rz} &= \sum_{\substack{z\in\{\widetilde{z_r},\ldots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} z f_{rz} + \overline{z} f_{r\overline{z}} + \underline{z} f_{r\underline{z}} + z^* f_{rz^*} \\
&= \sum_{\substack{z\in\{\widetilde{z_r},\ldots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} z f^*_{rz} + \overline{z} f^*_{r\overline{z}} - \frac{\overline{z}\epsilon}{\overline{z}-z^*} + \underline{z} f^*_{r\underline{z}} - \frac{\underline{z}\epsilon}{z^*-\underline{z}} + z^* f^*_{rz^*} + \frac{z^*\epsilon}{\overline{z}-z^*} + \frac{z^*\epsilon}{z^*-\underline{z}} \\
&= \sum_{z\in\{\widetilde{z_r},\ldots,Z\}} z f^*_{rz} - \epsilon + \epsilon \\
&= \sum_{z\in\{\widetilde{z_r},\ldots,Z\}} z f^*_{rz} = z^*.
\end{aligned}$$

so $(d^*, f) \in \overline{\mathcal{L}}(w)$. Consequently, by optimality of (d^*, f^*), we have

$$\sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f_{rz} \geqslant \sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f^*_{rz}.$$

However,

$$\begin{aligned}
\sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f_{rz} &= \sum_{\substack{z\in\{\widetilde{z_r},\ldots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} \beta_z f_{rz} + \beta_{\overline{z}} f_{r\overline{z}} + \beta_{\underline{z}} f_{r\underline{z}} + \beta_{z^*} f_{rz^*} \\
&= \sum_{\substack{z\in\{\widetilde{z_r},\ldots,Z\}\\ z\neq\overline{z},\underline{z},z^*}} \beta_z f^*_{rz} + \beta_{\overline{z}} f^*_{r\overline{z}} - \frac{\epsilon}{\overline{z}-z^*}\beta_{\overline{z}} + \beta_{\underline{z}} f^*_{r\underline{z}} - \frac{\epsilon}{z^*-\underline{z}}\beta_{\underline{z}} + \beta_{z^*} f^*_{rz^*} + \left(\frac{\epsilon}{\overline{z}-z^*} + \frac{\epsilon}{z^*-\underline{z}}\right)\beta_{z^*} \\
&= \sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f^*_{rz} - \frac{\epsilon\left(\overline{z}-\underline{z}\right)}{\left(\overline{z}-z^*\right)\left(z^*-\underline{z}\right)}\left(\frac{z^*-\underline{z}}{\overline{z}-\underline{z}}\beta_{\overline{z}} + \frac{\overline{z}-z^*}{\overline{z}-\underline{z}}\beta_{\underline{z}} - \beta_{z^*}\right).
\end{aligned}$$

Since γ verifies Condition (A), $z^*, \overline{z}, \underline{z} \in \{0, \ldots, Z\}$ and $\underline{z} < z^* < \overline{z}$, then using Lemma 12, we have

$$\frac{z^*-\underline{z}}{\overline{z}-\underline{z}}\beta_{\overline{z}} + \frac{\overline{z}-z^*}{\overline{z}-\underline{z}}\beta_{\underline{z}} > \beta_{z^*}.$$

Therefore, we have

$$\sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f_{rz} < \sum_{z\in\{\widetilde{z_r},\ldots,Z\}} \beta_z f^*_{rz}$$

which leads to a contradiction and concludes the proof of Theorem 4. □

C.3 Speed up of $\Lambda(v)$

Recall that $\Lambda(v)$ corresponds to

$$\min_{(w,D)} \left(c^T D\right) \qquad \text{Equation (5.20)}$$

$$s.t. \begin{cases} D \in \mathcal{D} & \text{Equations (5.7)-(5.12)} \\ (w, D) \in \mathcal{L} & \text{Equations (5.13)-(5.14)} \\ \sum_{s \in L_n} w_{ns\kappa_n} = 1 & \forall\ n \in \{1, \ldots, \overline{N}\} \end{cases}$$

As we mention in Section 5.4, there are several variables that we can set up to 0. Among these, we have the four following ones:

$$\forall\ n, k \in \{1, \ldots, \overline{N}\} \text{ s.t. } k \neq \kappa_n, \text{ then } \forall\ s \in L_n, w_{nsk} = 0.$$

$$\text{Let } n\{1, \ldots, \overline{N}\} \text{ s.t. } \kappa_n = 1, \text{ then } \forall\ s \in \mathcal{S}^{(E)} \setminus L_n,\ d_s^i = 0.$$

$$\forall\ k \in \{1, \ldots, \overline{N}\} \text{ s.t. } \kappa_n = k, \text{ then } \forall\ s \in \mathcal{S}^{(L)} \setminus L_n, \forall\ r \in \mathcal{S}^{(E)} \setminus E_n,\ d_{srk}^{(L)} = 0.$$

$$\forall\ k \in \{2, \ldots, \overline{N}\} \text{ s.t. } \kappa_n = k - 1 \text{ and } \kappa_m = k, \text{ then } \forall\ r \in \mathcal{S}^{(E)} \setminus E_n, \forall\ s \in \mathcal{S}^{(L)} \setminus L_m,\ d_{rsk}^{(E)} = 0.$$