Yard Crane Scheduling for container storage, retrieval, and relocation

Yard crane scheduling for container storage, retrieval, and relocation

Virgile Galle[*,a] Cynthia Barnhart[a,b] Patrick Jaillet[a,c]

vgalle@mit.edu cbarnhar@mit.edu jaillet@mit.edu

[*]Corresponding author, vgalle@mit.edu

[a]Operations Research Center, MIT, 77 Massachusetts Ave, Cambridge, MA 02139, USA.

[b]Civil & Environmental Engineering, MIT, 77 Massachusetts Ave, Cambridge, MA 02139, USA.

[c]Electrical Engineering & Computer Science, MIT, 77 Massachusetts Ave, Cambridge, MA 02139, USA.

Yard Crane Scheduling for container storage, retrieval, and relocation

Abstract

This paper introduces a novel optimization problem resulting from the combination of two major existing problems arising at storage yards in container terminals. The Yard Crane Scheduling Problem is typically concerned with routing the crane given a sequence of storage and retrieval requests to perform, while the Container Relocation Problem tackles the minimization of relocations when retrieving containers in a simpler setting. This paper is the first to consider a model that integrates these two problems by 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 and the linear programming relaxation of subproblems, we propose a heuristic local search scheme. Finally, we show the value of our solutions on both simulated instances as well as real data from a port terminal.

Keywords. Combinatorial optimization; OR in maritime industry; Integer programming; Yard Crane Scheduling Problem; Container relocation problem

1 Introduction

Container terminals, where containers are transferred between different modes of transportation both on the sea side and land side, 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 of both sea and land operations. At a typical container terminal, sea-side operations include assigning ships to quay slots or discharging and loading ships with quay cranes while land-side operations mostly involve routing internal trucks or storing and delivering containers in the storage area. For both types of operations, the efficiency of container terminals can be clearly enhanced by investments in new terminal devices (see Speer and Fischer (2017)). These investments can range from just improving current devices to buying new state-of-the-art cranes. Another direction for improvement is to develop new techniques to operate more efficiently existing devices, thus explaining the increasing research interests in optimizing operations in container terminals (see Gharehgozli et al. (2016)). This paper focuses on improving the efficiency of the storage and delivery of containers in the storage area through combinatorial optimization.

Due to limited space in the storage area, containers are stacked on top of each other. The resulting stacks create blocks of containers as shown in Figure 1. If a container that needs to be retrieved is not located at the top of its stack, that is, it is covered by other containers, the blocking containers must be relocated to another stack. As a result, during regular operations, one or more relocation moves are performed by the yard cranes. Such relocations (also called reshuffles), which cannot be charged to customers, create delays in operations, thus resulting in a substantial loss of revenue. Therefore, while this block structure represents a gain in space, it results in a loss in operational efficiency.

As more thoroughly explained in Section 3, two problems have been studied separately in the literature. One, the Container Relocation Problem, is concerned with minimizing the number of relocations given a sequence of requests in a restricted setting, most of the time a single bay ($Y = 1$ in Figure 1). The other, the Yard Crane Scheduling Problem, focuses on the routing of the crane and scheduling of storage and retrieval requests given space assignments for storage and relocations. This paper bridges the gap between these two problems in a unified framework. To the best of the our knowledge, this paper is the first to consider jointly these problems and to show the important benefits of jointly considering these decisions. We model this new problem using a binary integer programming formulation and study some properties of the problem. In practice, this problem has to be solved in real-time (a few minutes before the actual operations occur). Hence, we provide a heuristic procedure to generate promising solutions in a limited amount of time (1 minute), thereby showing the practical relevance and applicability of our approach.

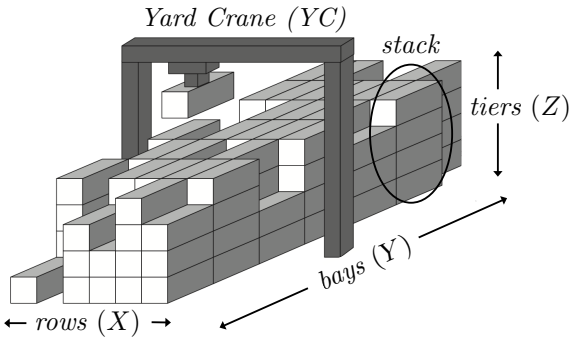


Figure 1: Container block with a single yard crane in port yard

The rest of our paper is organized as follows: After describing the problem of interest in Section 2, Section 3 explains the contributions of this work in light of an extensive literature review of both the Yard Crane Scheduling Problem and the Container Relocation Problem. Subsequently, Section 4 formulates the problem as a binary integer program and states some properties about this mathematical formulation. Section 5 builds upon these results to introduce a practical heuristic procedure. Algorithms are tested through computational experiments in Section 6. Finally, concluding remarks and future directions are given in Section 7.

2 Problem description

This section describes the problem of interest in this paper. Appendix A summarizes all assumptions and notations defined in this section.

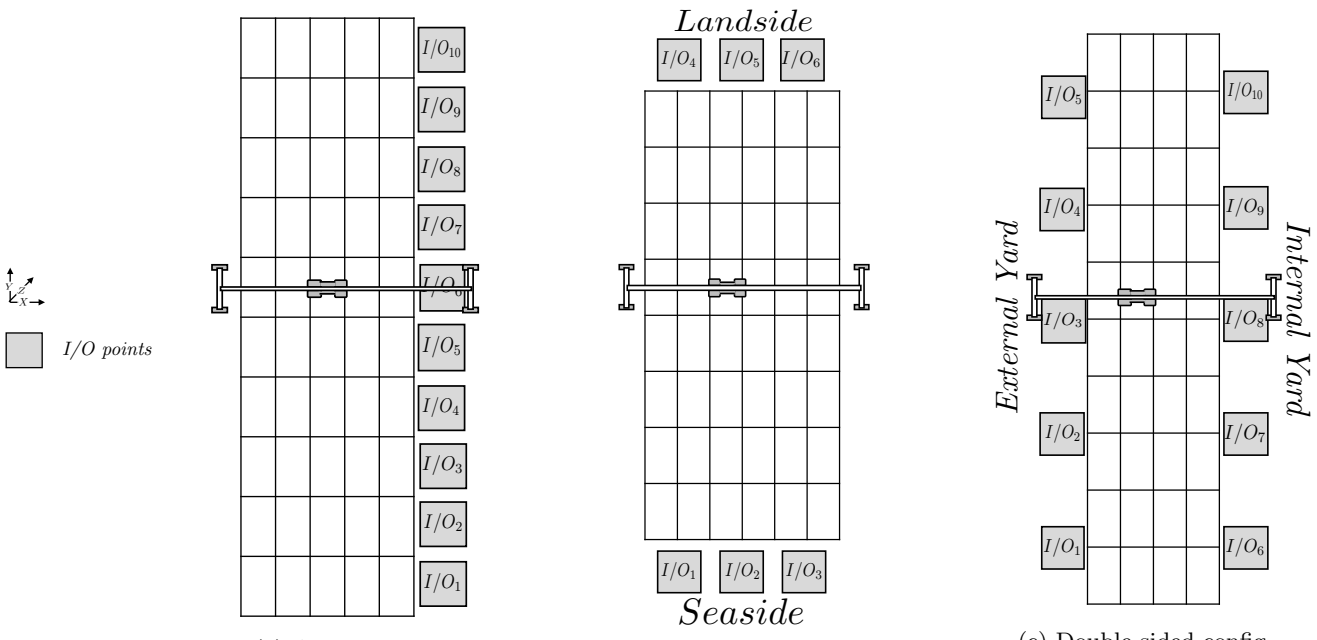
2.1 Problem geometry

We consider the following situation. A block consists of X rows, Y bays and Z tiers (see Figure 1) and we assume that this block is served by a single yard crane (YC), as shown in Figure 1. 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 $m \in \{1, \dots, M\}$. 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 configurations of I/O points. In the case of port yards, Wiese et al. (2010) and Carlo et al. (2014) discuss the two most frequent configurations (see Figures 2a and 2b), which are commonly referred to as Asian and European style configurations. The "double-sided" configuration is another configuration of interest (see Figure 2c) 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 $z_s^i \in \{0, \dots, Z\}$. Figure 4 shows these numbers in an example with Asian configuration. Note that these numbers are not given in Figure 2 just for the sake of clarity.



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

Figure 2: A top view of a block with three different I/O points configurations.

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 Carlo et al. (2014) for more details on handling equipment). In this paper, 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 Gharehgozli et al. (2014) and Yuan and Tang (2017)). The travel time of a YC is thoroughly studied in Speer and Fischer (2017) 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 3 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 (2017), it is important to consider the time to size the spreader during empty drives. However, we disregard this time because it is only required if there are different types of containers in the block, which is not the case in this paper. 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.

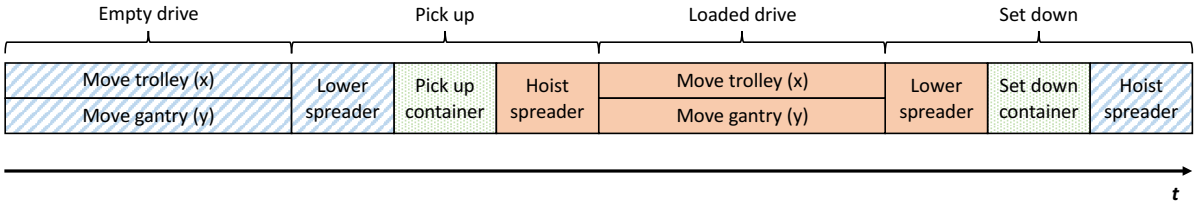


Figure 3: Pattern of typical YC movements for a given cycle. Striped blue indicates empty movements, solid red loaded movements and dotted green handling movements.

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 Gharehgozli et al. (2014). 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.

2.2 Requests

Given the problem geometry, the goal is to schedule storage and retrieval requests (also called productive requests or productive moves) by operating the YC in a minimum amount of time. As this process is dynamic, the block sequencing approach is typically adopted in order to consider a more static process (see Dell et al. (2009),Vis and Roodbergen (2009), Gharehgozli et al. (2014), Speer and Fischer (2017), Yuan and Tang (2017)). 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 Vis and Roodbergen (2009)). 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 in order 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, in order 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 Vis and Roodbergen (2009) or Gharehgozli et al. (2014).

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 $z_n \in \{1, \ldots, z_{s_n}^i\}$. If container n is on the top of its stack, *i.e.*, $z_n = z_{s_n}^i$, then it can be retrieved directly. However, for a significant number of requests, container n is blocked by other containers, *i.e.*, $z_n < z_{s_n}^i$, 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 Gharehgozli et al. (2016)).

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 Yuan and Tang (2017), 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. In order to ensure 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 4 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 containers $(z_i^s)_{s \in \mathcal{S}_B}$. 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 4. 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 in order to perform all these requests and assign new stacks to stored and relocated

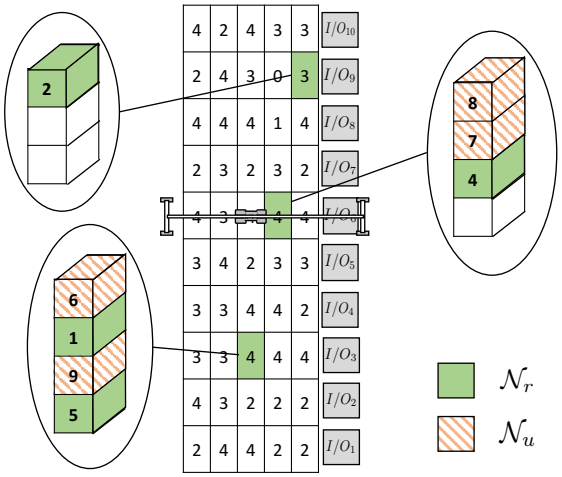


Figure 4: A top view of a block with Asian configuration. The integer in each stack of the block corresponds to the number of containers currently stored in the stack. For stacks in $\mathcal{S}_R$, we highlight containers to be retrieved ($\mathcal{N}_r$) and relocated ($\mathcal{N}_u$).

containers.

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 in order 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 Galle et al. (2016)). 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 Galle et al. (2017b)). 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 Galle et al. (2016), we have $\alpha_0 = 0$ and

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

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

$$\textit{cost-to-go} = \sum_{s \in \mathcal{S}_B} \alpha_{z_s^f}. \tag{2}$$

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

scalarization technique, by minimizing

$$\textit{Objective function} = \textit{immediate cost} + \gamma \times \textit{cost-to-go}, \tag{3}$$

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.

3 Literature review and contributions

This section first reviews previous work in Yard Crane Scheduling (YCS). For the reasons we mention in the introduction section, this topic has captured increasing attention over the last decade (see reviews from Lehnfeld and Knust (2014), Carlo et al. (2014) and Gharehgozli et al. (2016)). We focus our review on YCS with a single crane and we briefly mention some related work dealing with multiple cranes. Then, works on the Container Relocation Problem (CRP) and its variants are reviewed. Finally, we explain how this paper and the problem introduced in the previous section contribute to practice by bridging the gap between these two problems.

3.1 Yard Crane Scheduling

The first model for the YCSP with a single crane only considers retrieval requests and neither storage nor relocations enforced by these retrievals. It assumes that containers of the same type are stored in the same bay and most works are based on the Asian configuration. The retrieval schedule is given by groups of containers and the goal is to minimize crane travel time through the bays (intra bays travel time is not taken into account). Several approaches were tried to solve this problem: Kim and Kim (1999) propose the first Mixed Integer Program (MIP). Narasimhan and Palekar (2002) 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 of Kim and Kim (2003) uses encoding and decoding procedures embedded in a neighborhood beam search. Later, Lee et al. (2007) consider the same problem for two blocks (one crane per block) and introduce a simulated annealing scheme.

Ng and Mak (2005) extend the previous problem by including storage requests but still without considering relocations. They assume that each request has fixed start/end locations (*i.e.* L_n and E_n are singleton) 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. (2011) suggest to use A* and RBA* with an admissible heuristic.

Vis and Roodbergen (2009) 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 in order 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 in polynomial time using dynamic programming to link rows together and optimal assignments to solve subproblems within each row.

Dell et al. (2009) is the first work 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 only considering 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. (2014) considers 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 (*i.e.*, E_n is a singleton for $n \in \mathcal{N}_r$). 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. (2017) 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 (2017) solve a similar problem to the one in this paper 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 only minimizing 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 (2017) 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, even though it is not the primary focus of this paper, 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 (2017) 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. (2015) propose a similar approach as that of Gharehgozli et al. (2014) 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 only be selected by a single storage request, which is not practical. Park et al. (2010) consider storage and relocation location assignment using heuristics for twin RMG. However, as in Yuan and Tang (2017), they only take into account the crane travel time to perform $\overline{N}$ requests and not future operations. For other related problems such as simulation based models or inter-block crane allocations, we refer the reader to Carlo et al. (2014) and Gharehgozli et al. (2016).

3.2 Container relocation problem

The Container Relocation Problem (CRP) (also known as the Block Relocation Problem) is concerned with finding a sequence of moves of containers within a single bay that minimizes the number of relocations needed to retrieve all containers, while respecting a given order of retrieval requests. This problem was first formulated by Kim and Hong (2006) in a dynamic programming model. The literature on the CRP can be decomposed in two approaches. The first one aims

to find the optimal solution. The three primary techniques involve IP (see Expósito-Izquierdo et al. (2015), Zehendner et al. (2015), Galle et al. (2017a)), branch-and-bound (Expósito-Izquierdo et al. (2015) and Tricoire et al. (2017)) and the A^* algorithm (see Zhu et al. (2012), Tanaka and Takii (2016) and Borjian et al. (2015a)). Since the problem is $\mathcal{NP}$-hard (Caserta et al. (2012)), the second approach of studies tackles the CRP via heuristics such as constructive heuristics (for *e.g.* Caserta et al. (2012)) or rake search (Tricoire et al. (2017)).

There are many variants to the CRP. Galle et al. (2017b) suggest decision tree based solution methods for a stochastic version of the CRP. In relation to the YCS, Ünlüyurt and Aydın (2012) consider the objective function to be the crane travel and handling times instead of the number of relocations. A B&B algorithms and two heuristics are investigated to solve this problem. Lee and Lee (2010) extends the previous idea and propose a heuristic approach to solve the Container Retrieval Problem. In this problem, all bays in the block are considered and the objective is the crane travel time. The main difference with the YCS with 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.

Closely related to this paper, 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 bay. The first work for the DCRP is Wan et al. (2009) and it assumes that the order of requests is given. They identify the optimal solution to empty a bay using an IP similar to the ones proposed in 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 (2013) 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 (2014) consider the same problem as Wan et al. (2009) where the arrival (departure) sequences of containers to (from) the yard-bay 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. (2015b) introduce a variant of the DCRP by considering a class of flexible service policies to make minor changes in the order of container retrievals (a class generalized by our flexibilities introduced in Section 2).

In conclusion, Lehnfeld and Knust (2014) provide a general review and classification survey of the existing literature on the CRP and other related problems such as stacking or pre-marshalling problems.

3.3 Contributions to the literature

Based on this literature review, 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 Section 2 such as the third dimension of the block, the crane travel time features, the order flexibility and the dynamic nature of information (CRP and DCRP literature).

Recently some works have started to integrate 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 Dell et al. (2009) and Yuan and Tang (2017) for single crane scheduling or Park et al. (2010) in the case of two cranes. However, both Dell et al. (2009) and Park et al. (2010) consider rule-based heuristic solutions with some strong dependence on parameters that are hard to set in real operations. If Yuan and Tang (2017) is 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 aforementioned I/O point configurations 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 mentioned, this is not practical in many applications where external customers are concerned as fairness becomes an issue. The flexibility model proposed in this paper is widely applicable and could be worth studying in other settings.

5. In the setting introduced in this paper, a solution method needs to answer two questions simultaneously: i) crane movements; and ii) storage and relocation locations assignment. In order 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 paper, 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 paper is the testing of the two solution methods both on randomly generated data and real data from a port terminal.

4 Binary integer program

This section presents an exact method to solve the YSC problem with storage and relocation location assignments. In order 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 2:

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

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

Finally, we are given:

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

4.1 Formulation

4.1.1 Variables

The mathematical formulation introduced in this paper 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 only receive a stored or relocated container 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.

4.1.2 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 (4)-(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:

$$\forall\, n \in \{1,\dots,\overline{N}\}, \quad \sum_{\substack{s\in L_n \\ k\in\{1,\dots,\overline{N}\}}} w_{nsk} \;= 1, \tag{4a}$$

$$\forall\, k \in \{1,\dots,\overline{N}\}, \quad \sum_{\substack{n\in\{1,\dots,\overline{N}\} \\ s\in L_n}} w_{nsk} \;= 1. \tag{4b}$$

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

$$\forall\, n \in \{n' \in \mathcal{N}_r \cup \mathcal{N}_u \mid b_{n'} \neq 0\},\ \forall\, k \in \{1,\dots,\overline{N}\}, \sum_{s\in L_n} w_{nsk} - \sum_{\substack{s\in L_{b_n} \\ k'\in\{1,\dots,k-1\}}} w_{b_n sk'} \leqslant 0, \tag{5}$$

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

Order constraints – Recall that by definition of the flexibility of a productive request $n \in \{1,\dots,N\}$, 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:

$$\begin{aligned}&\forall\, n \in \{1,\dots,N\},\ \forall\, k \in \{n+\delta_n^+ + 1,\dots,\overline{N}\}, \\ &\qquad \sum_{\substack{n'\in\{1,\dots,N\} \\ s'\in L_{n'} \\ k'\in\{1,\dots,k-1\}}} w_{n's'k'} + \big(k - \big(n+\delta_n^+\big)\big) \times \sum_{s\in L_n} w_{nsk} \;\leqslant k-1,\end{aligned} \tag{6a}$$

$$\begin{aligned}&\forall\, n \in \{1,\dots,N\},\ \forall\, k \in \{1,\dots,\overline{N} - (N-n+\delta_n^-) - 1\}, \\ &\qquad \sum_{\substack{n'\in\{1,\dots,N\} \\ s'\in L_{n'} \\ k'\in\{k+1,\dots,\overline{N}\}}} w_{n's'k'} + \big(\overline{N} - k - \big(N-n+\delta_n^-\big)\big) \times \sum_{s\in L_n} w_{nsk} \;\leqslant \overline{N} - k.\end{aligned} \tag{6b}$$

The polyhedron $\mathcal{D}$ This polyhedron is defined by six types of constraints. We say $D \in \mathcal{D}$ if it verifies Equations (7)-(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):

$$\sum_{s\in S^{(L)}} d_s^i \;= 1, \tag{7a}$$

$$\forall\, k \in \{2,\dots,\overline{N}\}, \quad \sum_{\substack{r\in\mathcal{S}^{(E)} \\ s\in\mathcal{S}^{(L)}}} d_{rsk}^{(E)} \;= 1. \tag{7b}$$

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

$$\forall\, k \in \{1,\dots,\overline{N}\}, \quad \sum_{\substack{s\in\mathcal{S}^{(L)} \\ r\in\mathcal{S}^{(E)}}} d_{srk}^{(L)} \;= 1. \tag{8}$$

"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:

$$\forall s \in \mathcal{S}^{(L)}, \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{sr1} - d^{i}_{s} = 0, \tag{9a}$$

$$\forall s \in \mathcal{S}^{(L)}, \ \forall\, k \in \{2, \ldots, \overline{N}\}, \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{srk} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{rsk} = 0. \tag{9b}$$

"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:

$$\forall r \in \mathcal{S}^{(E)}, \ \forall\, k \in \{2, \ldots, \overline{N}\}, \sum_{s \in \mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = 0. \tag{10}$$

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

$$\forall\, r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1. \tag{11}$$

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,

$$\forall\, r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1, \ldots, \overline{N}\}}} d^{(L)}_{srk} = \widetilde{z}_r. \tag{12}$$

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 (13)-(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:

$$\forall\, n \in \{1, \ldots, \overline{N}\}, \ \forall\, s \in L_n, \ \forall\, k \in \{1, \ldots, \overline{N}\}, \sum_{r \in E_n} d^{(L)}_{srk} - w_{nsk} \geqslant 0. \tag{13}$$

Precedence relation between last retrieval and storage/relocation location assignments – Based on our assumption in Section 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):

$$\forall\, r \in \mathcal{S}_R, \ \forall\, k \in \{1, \ldots, \overline{N}\}, \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k' \in \{1, \ldots, k-1\}} w_{m_r r k'} \leqslant 0. \tag{14}$$

4.1.3 Objective function

We now formulate Equation (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:

$$\sum_{\substack{r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B \\ z \in \{\widetilde{z}_r, \ldots, Z\}}} \alpha_z f_{rz}, \tag{15}$$

where α_z is defined in Equation (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 3, 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_r^i. By definition, the empty first drive starts at stack s^i such that the cost of the first empty drive is given by

$$\sum_{r \in \mathcal{S}^{(L)}} t_{s^i r}^{(E)} d_r^i. \tag{16}$$

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

$$\sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)} \\ k \in \{2, \ldots, \overline{N}\}}} t_{rs}^{(E)} d_{rsk}^{(E)}. \tag{17}$$

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

$$\sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)} \\ k \in \{1, \ldots, \overline{N}\}}} t_{sr}^{(L)} d_{srk}^{(L)}. \tag{18}$$

Pick-ups and put-downs – As we pointed out in Section 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 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 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

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

Recall that we have $t^{(H)}(v(n)) = \dfrac{2(Z+1-v(n))}{v^z} + t^h$. Therefore, there is a constant part computed as $C = \sum_{n \in \mathcal{N}_r \cup \mathcal{N}_u} t^{(H)}(z_n) +$

$|\mathcal{N}_r \cup \mathcal{N}_s| \times t^{(H)}(1) + |\mathcal{N}_s \cup \mathcal{N}_u| \times t^{(H)}(0)$, 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 1. *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 1, we add the second term to the constant to get $C' = C + \frac{1}{v^z} \sum_{r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B} \widetilde{z}_r \left(\widetilde{z}_r + 1\right)$. 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:

$$-\sum_{\substack{r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B \\ z \in \{\widetilde{z}_r, \dots, Z\}}} \frac{1}{v^z} z\,(z+1)\, f_{rz}. \tag{19}$$

Combining Equations (16)-(19), we can write the objective function as

$$\sum_{s \in \mathcal{S}^{(L)}} t_{s^i s}^{(E)} d_s^i + \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)} \\ k \in \{2, \dots, \overline{N}\}}} t_{rs}^{(E)} d_{rsk}^{(E)} + \sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)} \\ k \in \{1, \dots, \overline{N}\}}} t_{sr}^{(L)} d_{srk}^{(L)} + \sum_{\substack{r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B \\ z \in \{\widetilde{z}_r, \dots, Z\}}} \beta_z f_{rz}, \tag{20}$$

where

$$\beta_z = \gamma \times \alpha_z - \frac{1}{v^z} z\,(z+1) = \gamma \times \left(z - \sum_{i=1}^{z} \frac{1}{i}\right) - \frac{1}{v^z} z\,(z+1). \tag{21}$$

Note that the objective function defined in Equation (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 paper, we say that γ verifies Condition (A) if

$$\gamma > \frac{2Z(Z-1)}{v^z}. \tag{22}$$

Lemma 2. *Let γ verify Condition (A), $z, z_1, z_2 \in \{0, \dots, Z\}$ such that $z_2 < z < z_1$, then we have*

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

4.1.4 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

$$\min_{(w,D)\in\{0,1\}} \left(c^T D\right) \qquad \text{Equation (20)}$$

$$s.t. \begin{cases} w \in \mathcal{W} & \text{Equations (4)-(6)} \\ D \in \mathcal{D} & \text{Equations (7)-(12)} \\ (w, D) \in \mathcal{L} & \text{Equations (13)-(14)} \end{cases}$$

4.2 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.

4.2.1 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. In order to do so, we define some notations:

$$\forall\ k \in \{1,\dots,\overline{N}\},\ (\nu_k, \sigma_k) = \left\{(n,s) \in \{1,\dots,\overline{N}\} \times \mathcal{S}^{(L)}\ \mid s \in L_n \text{ and } w_{nsk} = 1\right\}. \tag{23}$$

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

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \overline{E}_{\nu_k} = \{r \in E_{\nu_k}\ \mid (r \notin \mathcal{S}_R)\ \ or\ \ (r \in \mathcal{S}_R \text{ and } \exists\ k' \in \{1,\dots,k-1\} \text{ s.t. } \nu_{k'} = m_r)\} \tag{24}$$

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

$$\overline{\mathcal{S}}_B = \mathcal{S}^{(E)} \cap \mathcal{S}_B \cap \bigcup_{k \in \{1,\dots,\overline{N}\}} \overline{E}_{\nu_k}, \tag{25}$$

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

$$\forall\ r \in \overline{\mathcal{S}}_B,\ \overline{K}_r = \left\{ k \in \{1,\dots,\overline{N}\} \ \middle|\ r \in \overline{E}_{\nu_k} \right\}, \tag{26}$$

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

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

where

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

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 3. *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):

$$
\begin{aligned}
& d_{\sigma_1} = 1, \\
& f_{r\widetilde{z}_r} = 1, && \forall\, r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B, \\
& d^{(E)}_{r\sigma_k k} = \overline{d}_{r,k-1}, && \forall\, k \in \{2, \ldots, \overline{N}\},\ \forall\, r \in \overline{E}_{\nu_{k-1}}, \\
& d^{(L)}_{\sigma_k r k} = \overline{d}_{rk}, && \forall\, k \in \{1, \ldots, \overline{N}\},\ \forall\, r \in \overline{E}_{\nu_k}, \\
& f_{rz} = \overline{f}_{rz}, && \forall\, r \in \overline{\mathcal{S}}_B,\ \forall\, z \in \{\widetilde{z}_r, \ldots, Z\}
\end{aligned}
\tag{27}
$$

4.2.2 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 1. *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 4 in Appendix).

□

Theorem 2. *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

$$
\left\{
\begin{array}{l}
\displaystyle\min_{0 \leqslant (\overline{d}, \overline{f}) \leqslant 1} \left(\sum_{\substack{k \in \{1,\ldots,\overline{N}\} \\ r \in \overline{E}_{\nu_k}}} \overline{t}_{rk} \overline{d}_{rk} + \sum_{\substack{r \in \overline{\mathcal{S}}_B \\ z \in \{\widetilde{z}_r, \ldots, Z\}}} \beta_z \overline{f}_{rz} \right) \\
\\
s.t. \left\{
\begin{array}{ll}
\forall\, k \in \{1, \ldots, \overline{N}\}, & \displaystyle\sum_{r \in \overline{E}_{\nu_k}} \overline{d}_{rk} = 1, \\
\forall\, r \in \overline{\mathcal{S}}_B, & \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} \overline{f}_{rz} = 1, \\
\forall\, r \in \overline{\mathcal{S}}_B, & \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z \overline{f}_{rz} - \sum_{k \in \overline{K}_r} \overline{d}_{rk} = \widetilde{z}_r,
\end{array}
\right.
\end{array}
\right.
\tag{28}
$$

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

$$\min_{w \in \mathcal{W} \cap \{0,1\}} \left(g(w) \right). \tag{29}$$

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 1 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 (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 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.

5.1 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:

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

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

$$\forall\, n \in \{n' \in \mathcal{N}_r \cup \mathcal{N}_u \mid b_{n'} \neq 0\},\ \forall\, k \in \{1,\dots,\overline{N}\}, v_{nk} - \sum_{k'\in\{1,\dots,k-1\}} v_{b_n k'} \;\leqslant 0, \tag{30}$$

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

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

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 (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

$$\min_{v\in\mathcal{V}\cap\{0,1\}} \left(\min_{\sigma\in L} (h(v,\sigma)) \right). \tag{31}$$

where $L = \bigotimes_{n\in\{1,\dots,\overline{N}\}} L_n$ 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 (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.

5.2 1^{st} stage: restricted sampling on L

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

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

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

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

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 4 where we fix the order of requests according

to v. More specifically, consider the linear program $\Lambda(v)$:

$$\min_{(w,D)} \left(c^T D\right) \qquad \text{Equation (20)}$$

$$s.t. \begin{cases} D \in \mathcal{D} & \text{Equations (7)-(12)} \\ (w, D) \in \mathcal{L} & \text{Equations (13)-(14)} \\ \sum_{s \in L_n} w_{ns\kappa_n} = 1 & \forall\ n \in \{1, \ldots, \overline{N}\} \end{cases}$$

where the last constraints ensure 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

$$L\left(v\right) = \left\{\sigma = (\sigma_1, \ldots, \sigma_{\overline{N}}) \in L \ \middle|\ w^\Lambda_{n\sigma_n\kappa_n} > 0,\ \forall\ n \in \{1, \ldots, \overline{N}\}\right\}. \tag{32}$$

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

$$\forall\ n \in \{1, \ldots, \overline{N}\}, \mathbb{P}\left[\sigma_n = s\right] = w^\Lambda_{ns\kappa_n} \quad \text{and} \quad \mathbb{P}\left[\sigma = (s_1, \ldots, s_{\overline{N}})\right] = \prod_n \mathbb{P}\left[\sigma_n = s_n\right] \tag{33}$$

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 (32) and a probability distribution on this subset of L from Equation (33).
3. Sample without replacement $R'_1 = \min\left\{R_1, |L\left(v'\right)|\right\}$ points from the aforementioned probability distribution over $L(v)$ (denoted by $\left(\sigma^1, \ldots, \sigma^{R'_1}\right)$) and compute

$$\psi\left(v\right) = \min_{r \in \{1, \ldots, R'_1\}} \left(h(v, \sigma^r)\right). \tag{34}$$

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

5.3 2^{nd} 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 1.

Note that the definition of neighborhood is given in the pseudocode of Algorithm 1 (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.

Algorithm 1 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 $\Lambda(v(i))$ and compute $L\left(v(i)\right)$ from Equation (32). Let $R_1' = \min\{R_1, |L\left(v(i)\right)|\}$;
    for $r = 1 \ldots, R_1'$ **do** Sample σ^r without replacement from Equation (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 (32). Let $R_1' = \min\{R_1, |L\left(v'\right)|\}$;
          for $r = 1 \ldots, R_1'$ **do** Sample σ^r without replacement from Equation (33);
          Compute $\psi\left(v'\right) = \min\limits_{r \in \{1,\ldots,R_1'\}} \{h(v', \sigma^r)\}$ 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 $\left(v^{RRS}, \sigma^{RRS}\right) = \operatorname{argmin}\limits_{i \in \{1,\ldots,R_2\}} \{h(v(i), \sigma(i))\}$;

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 matching between requests and crane drives. Based on studies to sample efficiently on complete matching (for *e.g.* in Huber (2006)), 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 (30)). The pseudocode of SAMPLE_$\mathcal{V}$() is given in Algorithm 2.

Algorithm 2 Sampling algorithm using Accept-and-reject

procedure (v) = SAMPLE_$\mathcal{V}$()
  $v = \{0\}^{\overline{N}^2}$, $U\left(1\right) = \{1, \ldots, \overline{N}\}$, $k = 0$ and $k^p = 1$;
  while $k < \overline{N}$ **do** Increment $k = k + 1$;
    $\mathcal{R}(k^p, k) = U\left(k\right) \cap (\{n \in \mathcal{N}_s \cup \mathcal{N}_r \ \mid n - \delta_n^- \leqslant k^p \leqslant n + \delta_n^+\} \cup \mathcal{N}_u)$;
    while $|\mathcal{R}(k^p, k)| > 0$ **do** Sample n uniformly from $\mathcal{R}(k^p, k)$ and consider $v' = v$ and $v'_{nk} = 1$;
      if v' satisfies precedence constraints in Equation (30) $\forall\ k' \in \{1, \ldots, k\}$ **then**
        $v = v'$, $U\left(k+1\right) = U\left(k\right) \setminus \{n\}$ 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;

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.

6 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 in order 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. All results and code are available online at https://github.com/vgalle/YCSP.

Our experiments measure the performance of the proposed solutions over the long run, *i.e.*, each instance considers a large number of productive requests denoted by P. In this case, a typical performance indicator for a given solution is defined as the total travel cost of the crane (all empty/loaded drives, pick-ups and put-downs) incurred by this solution to perform all P productive requests, divided by P. This indicator represents the average cost of the given solution to perform a productive request (including the cost of associated unproductive requests) over the long run.

Recall that this paper takes the block sequencing approach (see Speer and Fischer (2017), Yuan and Tang (2017)), where a set of N "urgent" productive requests is considered and optimized before taking into account future productive requests. Thus, in a single instance, we solve a sequence of optimization problems until all P productive requests have been scheduled.

Consequently, each instance is defined by an initial block configuration, a sequence of P 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 P.

Moreover, 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.

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 A. Results for the heuristic method are reported for $(R_1, R_2) = (40, 40)$.

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 Caserta et al. (2012)). 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.

6.1 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.

6.1.1 Simulation parameters

In order 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 $P = 1500$ 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 only be retrieved 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 $N = 5$. In order 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). 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 $(0, 2)$.

6.1.2 Performance of different algorithms

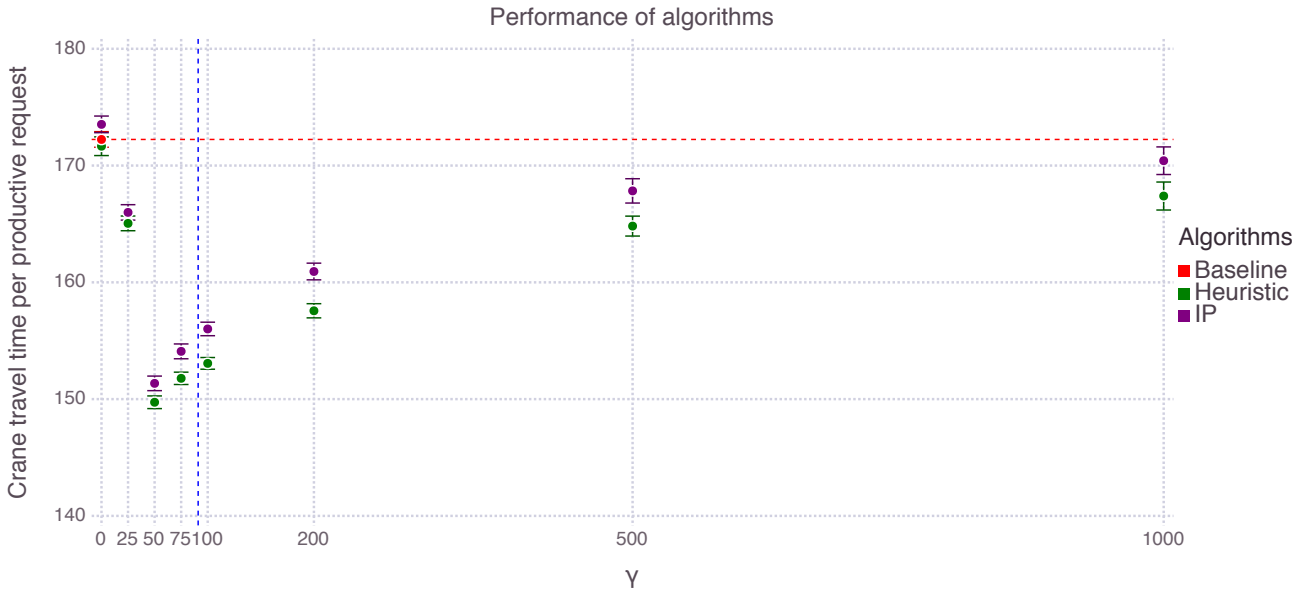


Figure 5: Performance of algorithms as function of γ: Each point represents the mean indicator obtained by different algorithms over the 30 randomly generated instances and the error bars represent $\pm 0.310 = \frac{1.699}{\sqrt{30}}$ standard deviations, corresponding to 95% confidence level. The red horizontal line corresponds to the mean of the baseline and the blue vertical line correspond to the lower bound on γ in Condition (A).

We compare the following algorithms:

- **Baseline**, the binary integer program introduced in section 4 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 $\delta = 2$.
- **IP**, the binary integer program introduced in section 4 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:

1. Figure 5 shows that increasing the weight on the cost-to-go can improve the solution. However, by putting too much weight on the cost-to-go, both the heuristic and the IP worsen as they neglect the immediate cost. Thus, Figure 5

Algorithm	Baseline	$\gamma = 0$		$\gamma = 25$		$\gamma = 50$		$\gamma = 75$		$\gamma = 100$		$\gamma = 200$		$\gamma = 500$		$\gamma = 1000$	
		IP	Heur.	IP	Heur.	IP	Heur.	IP	Heur.	IP	Heur.	IP	Heur.	IP	Heur.	IP	Heur.
Mean	15.04	15.9	14.65	10.87	10.24	1.08	0.0	2.91	1.37	4.2	2.23	7.48	5.24	12.09	10.08	13.81	11.79
Standard dev.	1.54	1.81	1.91	1.72	1.54	0.94	0.0	0.8	0.7	0.84	0.81	1.25	1.08	1.89	1.65	2.1	2.03
T-statistic	53.428	48.258	42.104	34.672	36.428	6.297	-	19.978	10.79	27.504	15.041	32.666	26.476	35.02	33.368	36.043	31.841

Table 2: Estimated statistics on 30 instances for the percent difference between each algorithm and the best-performing algorithm (heuristic with $\gamma = 50$). One can say that the best-performing algorithm improves over a given algorithm with a 95% confidence level if the T-statistic is greater than 1.699.

suggests that $\gamma = 50$ is the best value for both the heuristic and the IP in the setting of our experiments. **From this point forward, we consider the best performing algorithm to be the heuristic with $\gamma = 50$.** As a side note, this γ does not verify condition (A) in our setting.

2. Concerning the potential improvement incurred by considering future relocations, Table 2 shows that the best performing heuristic improves the baseline by more than 15%. Most importantly, choosing the right γ leads to a statistically significant improvement as all T-statistics are greater than the 95% confidence bound (1.699).

3. 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 as their 95% confidence intervals are overlapping). This demonstrates the value of using the heuristic over the IP even when $N = 5$. Table 3 reports the percentage of optimization problems that are not proven to be solved optimally by the IP, averaged over all 30 instances. There are many cases as such, 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 3: Percentage of optimization problems not proven to be solved optimally by the IP in the practical time limit of 60 seconds.

6.1.3 Impact of parameter δ

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 6, 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 (as the 95% confidence intervals are overlapping) and is relatively small compared to the benefit of a well tuned γ (see Figure 5). As the flexibility level solely depends on the port operator's policy, this latter can be set in order to balance the crane's efficiency (quantified in Figure 6) 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.

6.1.4 Impact of parameter N

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 7 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, 5% confidence intervals show that this positive impact is not statistically significant. We also

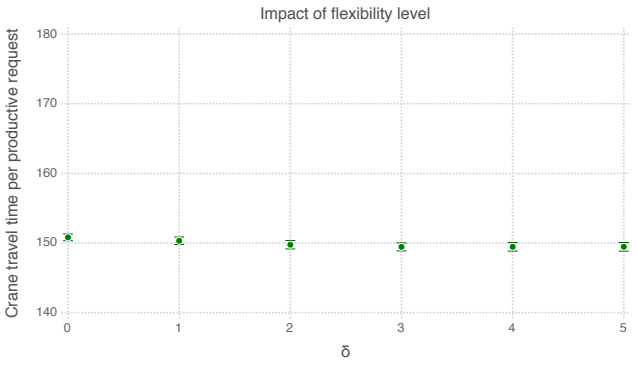


Figure 6: Impact of δ: Each point represents the mean indicator obtained by the heuristic with $\gamma = 50$ over all 30 instances and the error bars represent ± 0.310 standard deviations.

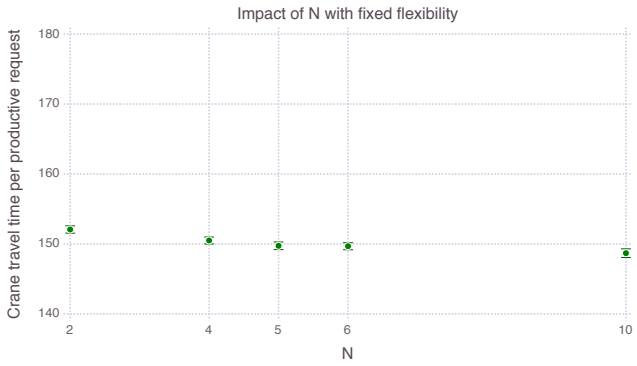


Figure 7: Impact of N: Each point represents the mean indicator obtained by the heuristic with $\gamma = 50$ over all 30 instances and the error bars represent ± 0.310 standard deviations.

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. Finally, since we limit the run time of the algorithm, it might affect the true impact of N.

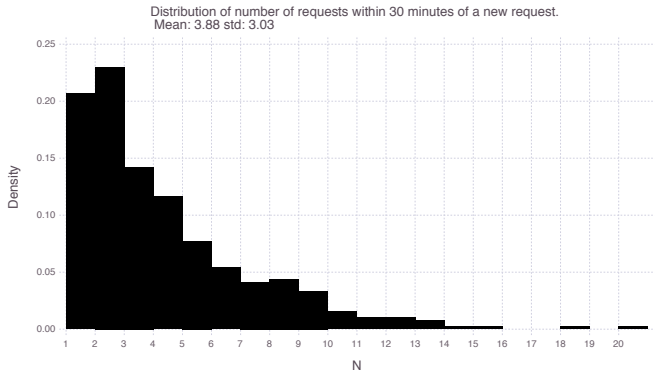
6.2 Data from a real terminal

6.2.1 Data processing

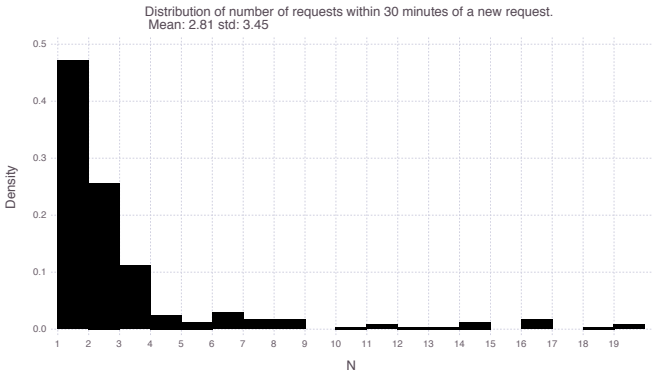
We collected data from two import blocks of a real port for 17 days in September 2017. The data included the position of each container in these two blocks on 09/07/2017 at 05:35:04 AM local time. For the next 17 days, each move of the cranes operating in these blocks was recorded. We summarize the main figures of this data set in Table 4. From this data, we can extract the initial block configuration as well as the sequence of productive requests.

Parameter	X	Y	Z	C	IO-Points	P = # 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 4: Data summary for requests in two blocks for 17 days in September 2017



8a Block 1



8b Block 2

Figure 8: Distribution of N of requests from two blocks for 17 days in September 2017

Recall that the third piece of data needed for our simulation is the sequence of Ns. In order 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 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 8 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 in order 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 A.

6.2.2 Results

Results are reported in Figure 9 for both blocks. These experiments provide three main insights:

1. Most importantly, using a good value for γ (here, 100) leads to an important 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.

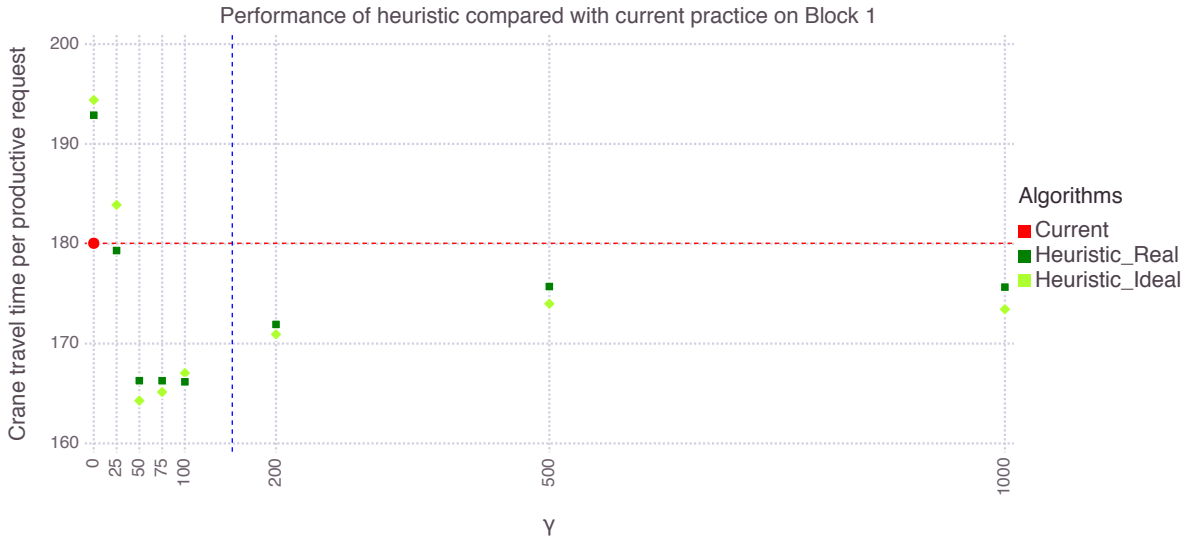
2. The performance of the heuristic as a function of γ in Block 1 are quite similar to the ones from the randomly generated instances, with the noticeable difference that the current practice is clearly outperforming the heuristic for $\gamma = 0$ (thus our artificial baseline which was even worse). 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 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 larger 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.

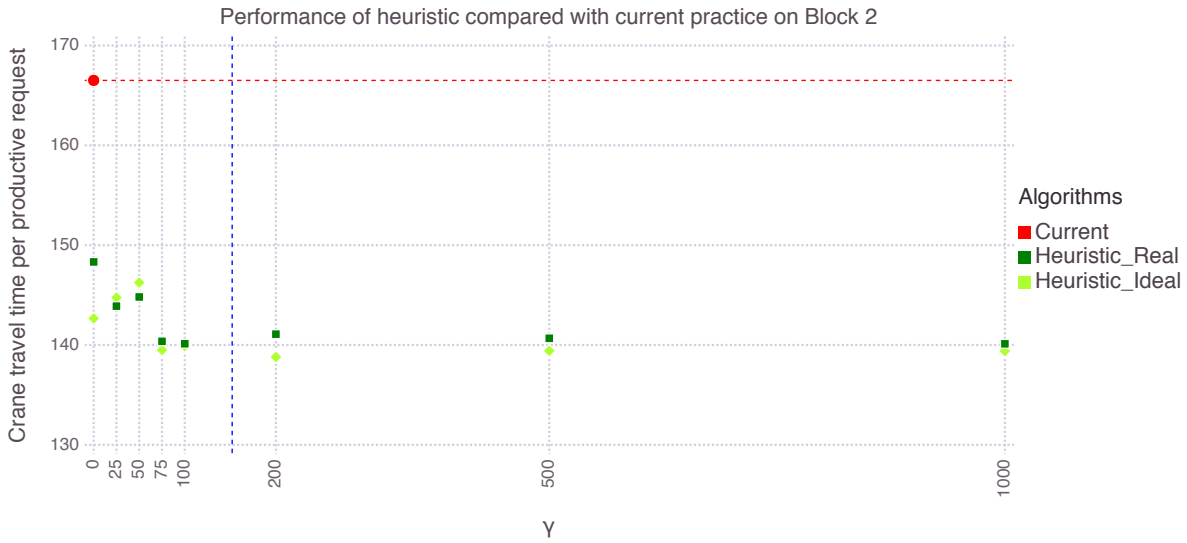
Note that the data only represents two particular instances and the conclusions we have drawn here might not apply in other specific cases.

6.3 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 important 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.



9a Block 1



9b Block 2

Figure 9: 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).

7 Conclusion

During the last decade, the increasing need for efficient systems at container terminals has led to the development of several operations research models. Due to the complexity of these models, researchers have mostly considered them in isolation, such as in the Yard Crane Scheduling Problem and the Container Relocation Problem. To the best of our knowledge, our paper is the first work that integrates both problems 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 randomly generated and data from a real terminal are conducted to show the efficiency and practicality of our heuristic method compared to the 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 look-ahead parameter γ).

Future research could find more efficient solutions for this new model and ways to automatically set γ based on simple business criteria. Important future work could consider our model with multiple crane systems in the case of ports dealing with crane interference (similar to Speer and Fischer (2017)). Another major challenge is the tactical allocation of incoming containers to blocks at the port level. Future research on this problem could optimize block allocation by modeling each block using our model.

References

Borjian, S., Galle, V., Manshadi, V. H., Barnhart, C., and Jaillet, P. (2015a). Container Relocation Problem: Approximation, Asymptotic, and Incomplete Information. *CoRR*, abs/1505.04229. (Accessed 11.01.2017).

Borjian, S., Manshadi, V. H., Barnhart, C., and Jaillet, P. (2015b). Managing Relocation and Delay in Container Terminals with Flexible Service Policies. *CoRR*, abs/1503.01535. (Accessed 11.01.2017).

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2):412–430.

Caserta, M., Schwarze, S., and Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1):96–104.

Dell, R. F., Royset, J. O., and Zyngiridis, I. (2009). Optimizing container movements using one and two automated stacking cranes. *Journal of Industrial and Management Optimization*, 5(2):285–302.

Expósito-Izquierdo, C., Melián-Batista, B., and Moreno-Vega, J. M. (2015). An exact approach for the Blocks Relocation Problem. *Expert Systems with Applications*, 42(17–18):6408 – 6422.

Galle, V., Barnhart, C., and Jaillet, P. (2017a). A new 0-1 formulation of the restricted container relocation problem based on a binary encoding of congurations. *submitted manuscript*. (Accessed 11.01.2017).

Galle, V., Borjian, S., Manshadi, V., Barnhart, C., and Jaillet, P. (2017b). The Stochastic Container Relocation Problem. *submitted manuscript*. (Accessed 11.01.2017).

Galle, V., Borjian Boroujeni, S., Manshadi, V., Barnhart, C., and Jaillet, P. (2016). An average-case asymptotic analysis of the Container Relocation Problem. *Operations Research Letters*, 44(6):723–728.

Gharehgozli, A. H., Laporte, G., Yu, Y., and de Koster, R. (2015). Scheduling Twin Yard Cranes in a Container Block. *Transportation Science*, 49(3):686–705.

Gharehgozli, A. H., Roy, D., and de Koster, R. (2016). Sea container terminals: New technologies and OR models. *Maritime Economics & Logistics*, 18(2):103–140.

Gharehgozli, A. H., Yu, Y., de Koster, R., and Udding, J. T. (2014). An exact method for scheduling a yard crane. *European Journal of Operational Research*, 235(2):431–447.

Gharehgozli, A. H., Yu, Y., Zhang, X., and de Koster, R. (2017). Polynomial Time Algorithms to Minimize Total Travel Time in a Two-Depot Automated Storage/Retrieval System. *Transportation Science*, 51(1):19–33.

Guo, X., Huang, S. Y., Hsu, W. J., and Low, M. Y. H. (2011). Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information. *Advanced Engineering Informatics*, 25(3):472–484.

Hakan Akyüz, M. and Lee, C.-Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Research Logistics (NRL)*, 61(2):101–118.

Huber, M. (2006). Exact Sampling from Perfect Matchings of Dense Regular Bipartite Graphs. *Algorithmica*, 44(3):183–193.

Kim, K. H. and Hong, G.-P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954.

Kim, K. H. and Kim, K. Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals.

Transportation Science, 33(1):17–33.

Kim, K. Y. and Kim, K. H. (2003). Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics (NRL)*, 50(5):498–514.

Lee, D.-H., Cao, Z., and Meng, Q. (2007). 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.

Lee, Y. and Lee, Y.-J. (2010). A heuristic for retrieving containers from a yard. *Computers & Operations Research*, 37(6):1139–1147.

Lehnfeld, J. and Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297–312.

Narasimhan, A. and Palekar, U. S. (2002). Analysis and algorithms for the transtainer routing problem in container port operations. *Transportation science*, 36(1):63–78.

Ng, W. and Mak, K. (2005). Yard crane scheduling in port container terminals. *Applied mathematical modelling*, 29(3):263–276.

Park, T., Choe, R., Ok, S. M., and Ryu, K. R. (2010). Real-time scheduling for twin RMGs in an automated container yard. *OR Spectrum*, 32(3):593–615.

Rei, R. and Pedroso, J. P. (2013). Tree search for the stacking problem. *Annals of Operations Research*, 203(1):371–388.

Speer, U. and Fischer, K. (2017). Scheduling of Different Automated Yard Crane Systems at Container Terminals. *Transportation Science*, 51(1):305–324.

Tanaka, S. and Takii, K. (2016). A Faster Branch-and-Bound Algorithm for the Block Relocation Problem. *IEEE Transactions on Automation Science and Engineering*, 13(1):181–190.

Tricoire, F., Fechter, J., and Beham, A. (2017). New insights on the block relocation problem. *Computers & Operations Research*.

Ünlüyurt, T. and Aydın, C. (2012). Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation*, 46(4):378–393.

Vis, I. F. and Roodbergen, K. J. (2009). Scheduling of container storage and retrieval. *Operations Research*, 57(2):456–467.

Wan, Y.-w., Liu, J., and Tsai, P.-C. (2009). The assignment of storage locations to containers for a container stack. *Naval Research Logistics (NRL)*, 56(8):699–713.

Wiese, J., Suhl, L., and Kliewer, N. (2010). Mathematical models and solution methods for optimal container terminal yard layouts. *OR Spectrum*, 32(3):427–452.

Yuan, Y. and Tang, L. (2017). 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.

Zehendner, E., Casserta, M., Feillet, D., Schwarze, S., and Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. *European Journal of Operational Research*, 245(2):415–422.

Zhu, W., Qin, H., Lim, A., and Zhang, H. (2012). Iterative Deepening A* Algorithms for the Container Relocation Problem. *IEEE Transactions on Automation Science and Engineering*, 9(4):710–722.

A Assumptions and notations summary

A.1 Assumptions

- Each block is considered as an independent entity which can only store a unique type of container and is limited in all three dimensions to (X, Y, Z) of such containers.
- A single yard crane (of any type, *i.e.*, RMG, TRG,...) operates in the considered block, hence without any interference with another crane.
- The model used to compute travel times of the yard crane is the one introduced by Speer and Fischer (2017) where spreader sizing is disregarded due to the unique type of containers stored in the block.
- Any configuration with m I/O points can be considered.
- In order to deal with the continuous nature of the problem, the classical block sequencing approach is taken: given the initial number of containers per stack and the initial position of the crane handler, we only consider a list of urgent productive requests (retrieval and storage), as well as the unproductive requests (relocations) implied by the aforementioned retrieval requests.
- The number of crane cycles in one optimization problem is exactly equal to the number of productive and unproductive requests. This implies that no container can be stacked or relocated before all containers have been retrieved from a given stack.
- Each request has a given order flexibility as well as location limitations in picking up and dropping off containers.
- First-Come-First-Serve policy is always assumed feasible. In order to do so, we assume that, if two containers are retrieved from the same stack, the above container should retrieved first.

A.2 Notations

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 5).

$\left(v^{y,E}, v^{y,L}\right)$: 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.

B Technical Proofs

Lemma 1. *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*

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

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 5: 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 Gharehgozli et al. (2014). No separating space between containers is considered.

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 2. *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

$$\beta_{z+1} + \beta_{z-1} > 2\beta_z. \tag{35}$$

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 (35). Using Equation (35) repeatedly, we have

$$\beta_{z_1} + (z_1 - z)\,\beta_{z-1} > (z_1 - z + 1)\,\beta_z \text{ and } (z - z_2)\,\beta_{z+1} + \beta_{z_2} > (z - z_2 + 1)\,\beta_z. \tag{36}$$

Consequently, we have

$$\begin{aligned}
\frac{z - z_2}{z_1 - z_2}\beta_{z_1} + \frac{z_1 - z}{z_1 - z_2}\beta_{z_2} &> \frac{(z - z_2)\,(z_1 - z + 1)}{z_1 - z_2}\beta_z - \frac{(z - z_2)\,(z_1 - z)}{z_1 - z_2}\beta_{z-1} \\
&\quad + \frac{(z_1 - z)\,(z - z_2 + 1)}{z_1 - z_2}\beta_z - \frac{(z_1 - z)\,(z - z_2)}{z_1 - z_2}\beta_{z+1} \\
&= \beta_z - \frac{(z_1 - z)\,(z - z_2)}{z_1 - z_2}\left(\beta_{z+1} + \beta_{z-1} - 2\beta_z\right) \\
&> \beta_z,
\end{aligned}$$

where the first inequality holds thanks to Equation (36) and the second holds with Equation (35) and $z_2 < z < z_1$. This concludes the proof of Lemma 2. □

Lemma 3. *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. In order 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:

$$\forall\, k \in \{1, \ldots, \overline{N}\},\ \begin{cases} \forall\, s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0, \\ \forall r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k r k} = 0. \end{cases} \tag{37}$$

$$\forall\, k \in \{1, \ldots, \overline{N}\},\ \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} = 1. \tag{38}$$

$$\begin{cases} d^i_{\sigma_1} = 1, \\ \forall\, s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\},\, d^i_s = 0. \end{cases} \tag{39}$$

$$\forall\, k \in \{2, \ldots, \overline{N}\},\ \forall\, r \in \mathcal{S}^{(E)},\ \forall s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ d^{(E)}_{rsk} = 0. \tag{40}$$

$$\forall\, k \in \{2, \ldots, \overline{N}\},\ \forall\, r \in \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} - d^{(L)}_{\sigma_{k-1}, r, k-1} = 0. \tag{41}$$

$$\forall\, k \in \{2, \ldots, \overline{N}\},\ \forall\, r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} = 0. \tag{42}$$

$$\forall r \in \overline{\mathcal{S}}_B,\ \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k} = \widetilde{z}_r. \tag{43}$$

$$\forall r \in \overline{\mathcal{S}}_B,\ \sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1, \tag{44}$$

$$\forall r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B,\ \begin{cases} f_{r\widetilde{z}_r} = 1, \\ \forall\, z \in \{\widetilde{z}_r + 1, \ldots, Z\},\, f_{rz} = 0. \end{cases} \tag{45}$$

Proof of implied Constraint (37) We first show that

$$\begin{gathered}\forall\ k \in \{1,\dots,\overline{N}\}, \forall\ s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0,\\ \forall\ k \in \{1,\dots,\overline{N}\},\ \forall r \in \mathcal{S}^{(E)} \setminus E_{\nu_k},\ d^{(L)}_{\sigma_k rk} = 0.\end{gathered} \tag{46}$$

Let $k \in \{1,\dots,\overline{N}\}$. By definition, we have $w_{\nu_k\sigma_k k} = 1$, so Constraint (13) for $n = \nu_k$ and $s = \sigma_k$ implies that

$$0 \leqslant \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k rk} - w_{\nu_k\sigma_k k} = \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k rk} - 1.$$

We combine this inequality with Constraint (8) which implies that

$$\begin{aligned}0 &= \sum_{\substack{s \in \mathcal{S}^{(L)}\\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} - 1\\ &= \sum_{\substack{s \in \mathcal{S}^{(L)}\setminus\{\sigma_k\}\\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} + \sum_{r \in \mathcal{S}^{(E)}\setminus E_{\nu_k}} d^{(L)}_{\sigma_k rk} + \sum_{r \in E_{\nu_k}} d^{(L)}_{\sigma_k rk} - 1\\ &\geqslant \sum_{\substack{s \in \mathcal{S}^{(L)}\setminus\{\sigma_k\}\\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} + \sum_{r \in \mathcal{S}^{(E)}\setminus E_{\nu_k}} d^{(L)}_{\sigma_k rk},\end{aligned}$$

Since $d^{(L)}_{srk}$ are non-negative, this proves Equation (46). We further improve this constraint by showing that

$$\forall\ k \in \{1,\dots,\overline{N}\}, \forall\ r \in E_{\nu_k} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k rk} = 0. \tag{47}$$

Let $k \in \{1,\dots,\overline{N}\}$ and $r \in E_{\nu_k} \setminus \overline{E}_{\nu_k}$. First, using Equation (46), we have $\sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} = d^{(L)}_{\sigma_k rk}$. 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 $\forall\ k' \in \{1,\dots,k-1\}$, $w_{m_r rk'} = 0$, thus $\sum_{k' \in \{1,\dots,k-1\}} w_{m_r rk'} = 0$. By using both these observations in Constraint (14), we have

$$0 \geqslant \sum_{s \in \mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k' \in \{1,\dots,k-1\}} w_{m_r rk'} = d^{(L)}_{\sigma_k rk}.$$

As $d^{(L)}_{\sigma_k rk}$ are non-negative, this proves Equation (47). Combined with Equation (46), this proves implied Constraint (37).

Proof of implied Constraint (38) Using implied Constraint (37) together with Constraint (8), we directly get implied Constraint (38).

Proof of implied Constraint (39) We have

$$d^i_{\sigma_1} = \sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_1 r1} = \sum_{r \in \overline{E}_{\nu_1}} d^{(L)}_{\sigma_1 r1} = 1,$$

where the first equality comes from Constraint (9a) for $s = \sigma_1$, the second from implied Constraint (37) and the last from implied Constraint (38) for $k = 1$. By combining this latter fact with Constraint (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 (39).

Proof of implied Constraint (40) Let $k \in \{2, \dots, \overline{N}\}$, 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 (9b) for $s = \sigma_k$, the second equality comes from implied Constraint (37) and the last from implied Constraint (38). Based on this observation, we use Constraint (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 (40).

Proof of implied Constraint (41) Let $k \in \{2, \dots, \overline{N}\}$ 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 (41). The first equality is Constraint (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 (40) and (37).

Proof of implied Constraint (42) Let $k \in \{2, \dots, \overline{N}\}$ and $r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}}$. Using both implied Constraints (41) and (37), we have

$$d^{(E)}_{r\sigma_k k} = d^{(L)}_{\sigma_{k-1},r,k-1} = 0,$$

proving implied Constraint (42).

Proof of implied Constraint (43) Let $r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B$. Using implied Constraint (37) and by definition of $\overline{K}_r$, we note that

$$\sum_{\substack{s \in \mathcal{S}^{(L)} \\ k \in \{1,\dots,\overline{N}\}}} d^{(L)}_{srk} = \sum_{k \in \{1,\dots,\overline{N}\}} d^{(L)}_{\sigma_k r k} = \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k}.$$

By replacing this expression in Constraint (12), we get

$$\forall\ r \in \mathcal{S}^{(E)} \cap \mathcal{S}_B, \quad \sum_{z \in \{\widetilde{z}_r, \dots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k} = \widetilde{z}_r. \tag{48}$$

Since $\overline{\mathcal{S}}_B \subset \mathcal{S}^{(E)} \cap \mathcal{S}_B$, this proves implied Constraint (43).

Proof of implied Constraint (44) This implied constraint is directly proven by Constraint (11) since $\overline{S}_B \subset \mathcal{S}^{(E)} \cap \mathcal{S}_B$.

Proof of implied Constraint (45) 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 (37) we have $\forall k \in \{1, \dots, \overline{N}\}$, $d^{(L)}_{\sigma_k r k} = 0$, thus

$$\sum_{k \in \{1,\dots,\overline{N}\}} d^{(L)}_{\sigma_k r k} = 0.$$

Consequently, we have

$$\begin{aligned}
0 &= \widetilde{z}_r - \sum_{z \in \{\widetilde{z}_r, \dots, Z\}} z f_{rz} + \sum_{k \in \{1, \dots, \overline{N}\}} d^{(L)}_{\sigma_k r k} \\
&= \widetilde{z}_r - \sum_{z \in \{\widetilde{z}_r, \dots, Z\}} z f_{rz} \\
&= \widetilde{z}_r (1 - f_{r\widetilde{z}_r}) + \sum_{z \in \{\widetilde{z}_r + 1, \dots, Z\}} z f_{rz} \\
&\geqslant \sum_{z \in \{\widetilde{z}_r + 1, \dots, Z\}} z f_{rz} \\
&\geqslant \sum_{z \in \{\widetilde{z}_r + 1, \dots, Z\}} f_{rz},
\end{aligned}$$

and since f_{rz} are non-negative, this proves that $\forall\, z \in \{\widetilde{z}_r + 1, \dots, Z\}$, $f_{rz} = 0$. We note that the first equality comes from Constraint (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 $z \in \{\widetilde{z}_r + 1, \dots, Z\}$. Finally, using this fact and Constraint (11), we have

$$1 = \sum_{z \in \{\widetilde{z}_r, \dots, Z\}} f_{rz} = f_{r\widetilde{z}_r},$$

which proves implied Constraint (45).

Redundancy of original constraints We now show that Constraints (7)-(14) are redundant with implied Constraints (37)-(45).

Constraint (7a) is redundant with implied Constraint (39):

$$\sum_{s \in S^{(L)}} d^i_s = d^i_{\sigma_1} = 1.$$

Constraint (7b) is redundant with implied Constraints (40), (42), (41) and (39):

$$\forall\, k \in \{2, \dots, \overline{N}\}, \sum_{\substack{r \in \mathcal{S}^{(E)} \\ s \in \mathcal{S}^{(L)}}} d^{(E)}_{rsk} = \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(E)}_{r\sigma_k k} = \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(L)}_{\sigma_{k-1}, r, k-1} = 1.$$

Constraint (8) is redundant with implied Constraints (37) and (39):

$$\forall\, k \in \{1, \dots, \overline{N}\}, \sum_{\substack{s \in \mathcal{S}^{(L)} \\ r \in \mathcal{S}^{(E)}}} d^{(L)}_{srk} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k, r, k} = 1.$$

Constraint (9a) is redundant with implied Constraints (37), (39) and (38):

$$\forall\, s \in \mathcal{S}^{(L)}, \begin{cases} \text{if } s = \sigma_1, & \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_1 r 1} - d^i_{\sigma_1} = \sum_{r \in \overline{E}_{\nu_1}} d^{(L)}_{\sigma_1 r 1} - 1 = 1 - 1 = 0. \\ \text{if } s \neq \sigma_1, & \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{sr1} - d^i_s = 0 - 0 = 0. \end{cases}$$

Constraint (9b) is redundant with implied Constraints (37), (40), (42) and (38):

$$\begin{matrix} \forall\, s \in \mathcal{S}^{(L)} \\ \forall\, k \in \{2, \dots, \overline{N}\} \end{matrix}, \begin{cases} \text{if } s = \sigma_k, & \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{\sigma_k r k} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{r\sigma_k k} = \sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} - \sum_{r \in \overline{E}_{\nu_{k-1}}} d^{(L)}_{\sigma_{k-1}, r, k-1} = 1 - 1 = 0. \\ \text{if } s \neq \sigma_k, & \displaystyle\sum_{r \in \mathcal{S}^{(E)}} d^{(L)}_{srk} - \sum_{r \in \mathcal{S}^{(E)}} d^{(E)}_{rsk} = 0 - 0 = 0. \end{cases}$$

Constraint (10) is redundant with implied Constraints (37), (40), (42) and (38):

$$\begin{array}{c}\forall\, r \in \mathcal{S}^{(E)} \\ \forall\, k \in \{2,\dots,\overline{N}\}\end{array}, \begin{cases} \text{if } r \in \overline{E}_{\nu_k}, & \displaystyle\sum_{s\in\mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s\in\mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = d^{(L)}_{\sigma_{k-1},r,k-1} - d^{(L)}_{\sigma_{k-1},r,k-1} = 0. \\ \text{if } r \notin \overline{E}_{\nu_k}, & \displaystyle\sum_{s\in\mathcal{S}^{(L)}} d^{(E)}_{rsk} - \sum_{s\in\mathcal{S}^{(L)}} d^{(L)}_{s,r,k-1} = d^{(E)}_{r\sigma_k k} - d^{(L)}_{\sigma_{k-1},r,k-1} = 0 - 0 = 0. \end{cases}$$

Constraint (11) is redundant with implied Constraints (44) and (45):

$$\forall\, r \in \mathcal{S}_B, \begin{cases} \text{if } r \in \overline{\mathcal{S}}_B, & \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f_{rz} = 1, \\ \text{if } r \in\notin \overline{\mathcal{S}}_B, & \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} f_{rz} = f_{r\widetilde{z}_r} = 1. \end{cases}$$

Constraint (12) is redundant with implied Constraints (37), (43) and (45):

$$\forall\, r \in \mathcal{S}_B, \begin{cases} \text{if } r \in \overline{\mathcal{S}}_B, & \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f_{rz} - \sum_{\substack{s\in\mathcal{S}^{(L)} \\ k\in\{1,\dots,\overline{N}\}}} d^{(L)}_{srk} = \sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f_{rz} - \sum_{k\in\overline{K}_r} d^{(L)}_{\sigma_k rk} = \widetilde{z}_r. \\ \text{if } r \notin \overline{\mathcal{S}}_B, & \displaystyle\sum_{z\in\{\widetilde{z}_r,\dots,Z\}} z f_{rz} - \sum_{\substack{s\in\mathcal{S}^{(L)} \\ k\in\{1,\dots,\overline{N}\}}} d^{(L)}_{srk} = \widetilde{z}_r - 0 = \widetilde{z}_r. \end{cases}$$

Constraint (13) is redundant with implied Constraints (37) and (38):

$$\begin{array}{c}\forall\, n \in \{1,\dots,\overline{N}\} \\ \forall\, s \in L_n \\ \forall\, k \in \{1,\dots,\overline{N}\}\end{array}, \begin{cases} \text{if } n = \nu_k \text{ and } s = \sigma_k, & \displaystyle\sum_{r\in E_{\nu_k}} d^{(L)}_{\sigma_k rk} - w_{\nu_k\sigma_k k} = \sum_{r\in\overline{E}_{\nu_k}} d^{(L)}_{\sigma_k rk} - 1 = 0 \geqslant 0. \\ \text{if } n \neq \nu_k \text{ or } s \neq \sigma_k, & \displaystyle\sum_{r\in E_n} d^{(L)}_{srk} - w_{nsk} = \sum_{r\in E_n} d^{(L)}_{srk} \geqslant 0. \end{cases}$$

Recall that by definition, if $r \in \mathcal{S}_R \cap \overline{E}_{\nu_k}$ then $\exists\, k' \in \{1,\dots,k-1\}$ such that $\nu_{k'} = m_r$, *i.e.*, $w_{m_r rk'} = 1$ so $\displaystyle\sum_{k'\in\{1,\dots,k-1\}} w_{m_r rk'} = 1$. Thus, Constraint (14) is redundant with implied Constraints (37):

$$\begin{array}{c}\forall\, r \in \mathcal{S}_R \\ \forall\, k \in \{1,\dots,\overline{N}\}\end{array}, \begin{cases} \text{if } r \in \overline{E}_{\nu_k}, & \displaystyle\sum_{s\in\mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k'\in\{1,\dots,k-1\}} w_{m_r rk'} \leqslant d^{(L)}_{\sigma_k rk} - 1 \leqslant 0. \\ \text{if } r \notin \overline{E}_{\nu_k}, & \displaystyle\sum_{s\in\mathcal{S}^{(L)}} d^{(L)}_{srk} - \sum_{k'\in\{1,\dots,k-1\}} w_{m_r rk'} = 0 - \sum_{k'\in\{1,\dots,k-1\}} w_{m_r rk'} \leqslant 0. \end{cases}$$

Thus $\Pi(w)$ is equivalent to

$$\min_{D\in\{0,1\}} \left(c^T D\right)$$
$$s.t.\ D \text{ satisfice Constraints (37)-(45)}$$

Equivalence of $\overline{\Pi}(w)$ and $\Pi(w)$ Among Constraints (37)-(45), some of them fix variables to 0 or 1. In summary, we have

$$\forall\, k \in \{1,\dots,\overline{N}\}, \begin{cases} \forall\, s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ \forall r \in \mathcal{S}^{(E)},\ d^{(L)}_{srk} = 0, \\ \forall r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_k},\ d^{(L)}_{\sigma_k rk} = 0. \end{cases}$$

$$\begin{cases} d^i_{\sigma_1} = 1, \\ \forall\, s \in \mathcal{S}^{(L)} \setminus \{\sigma_1\}, d^i_s = 0. \end{cases}$$

$$\forall\, k \in \{2,\dots,\overline{N}\}, \begin{cases} \forall\, r \in \mathcal{S}^{(E)},\ \forall s \in \mathcal{S}^{(L)} \setminus \{\sigma_k\},\ d^{(E)}_{rsk} = 0, \\ \forall\, r \in \mathcal{S}^{(E)} \setminus \overline{E}_{\nu_{k-1}},\ d^{(E)}_{r\sigma_k k} = 0, \end{cases}$$

$$\forall r \in \left(\mathcal{S}^{(E)} \cap \mathcal{S}_B\right) \setminus \overline{\mathcal{S}}_B, \begin{cases} f_{r\widetilde{z}_r} = 1, \\ \forall\ z \in \{\widetilde{z}_r + 1, \ldots, Z\}, f_{rz} = 0. \end{cases}$$

These constraints and their associated variables can be treated as constant for the subproblem $\Pi(w)$. 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 (41)), 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

$$\begin{cases} \min \left(\displaystyle\sum_{\substack{k \in \{1,\ldots,\overline{N}-1\} \\ r \in \overline{E}_{\nu_k}}} \left(t^{(E)}_{r\sigma_{k+1}} + t^{(L)}_{\sigma_k r}\right) d^{(L)}_{\sigma_k r k} + \sum_{\substack{k = \overline{N} \\ r \in \overline{E}_{\nu_k}}} t^{(L)}_{\sigma_k r} d^{(L)}_{\sigma_k r k} + \sum_{\substack{r \in \overline{\mathcal{S}}_B \\ z \in \{\widetilde{z}_r, \ldots, Z\}}} \beta_z f_{rz} \right) \\ \\ s.t. \begin{cases} \forall\ k \in \{1, \ldots, \overline{N}\},\ \displaystyle\sum_{r \in \overline{E}_{\nu_k}} d^{(L)}_{\sigma_k r k} = 1, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} z f_{rz} - \sum_{k \in \overline{K}_r} d^{(L)}_{\sigma_k r k} = \widetilde{z}_r, \\ \forall\ r \in \overline{\mathcal{S}}_B,\ \displaystyle\sum_{z \in \{\widetilde{z}_r, \ldots, Z\}} f_{rz} = 1. \end{cases} \end{cases}$$

By changing the variables of $\Pi_2(w)$ such that

$$\forall\ k \in \{1, \ldots, \overline{N}\},\ \forall\ r \in \overline{E}_{\nu_k},\ \overline{d}_{rk} = d^{(L)}_{\sigma_k r k},$$

and

$$\forall\ s \in \overline{\mathcal{S}}_B,\ \forall\ z \in \{\widetilde{z}_r, \ldots, Z\},\ \overline{f}_{rz} = f_{rz},$$

and by definition of $\bar{t}_{rk}$, 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 1. *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 $D^1 = \left(d^1, f^*\right)$ and $D^2 = \left(d^2, f^*\right)$ such that

$$\begin{aligned} d^1_{pl} &= d^*_{pl} - \epsilon & \qquad d^2_{pl} &= d^*_{pl} + \epsilon \\ d^1_{ql} &= d^*_{ql} + \epsilon & \qquad d^2_{ql} &= d^*_{ql} - \epsilon \end{aligned}$$

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 $\forall r \in \overline{\mathcal{S}}_B$, $\sum_{k \in \{1,\dots,\overline{N}\}} d^1_{rk} = \sum_{k \in \{1,\dots,\overline{N}\}} d^2_{rk} = \sum_{k \in \{1,\dots,\overline{N}\}} d^*_{rk}$. 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 $d^* = \frac{1}{2}\left(d^1 + d^2\right)$ so $D^* = \frac{1}{2}\left(D^1 + D^2\right)$, 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 1 under this case requires a technical lemma (Lemma 4). Let us define

$$\forall\ k \in \{1,\dots,\overline{N}\},\ \mathcal{R}_k = \left\{r \in \overline{E}_{\nu_k} \mid d^*_{rk} \notin \{0,1\}\right\} \text{ and } \mathcal{R} = \bigcup_{k \in \{1,\dots,\overline{N}\}} \mathcal{R}_k.$$

By definition, we have $p, q \in \mathcal{R}_l$ so $|\mathcal{R}| > 0$. We also define

$$\forall\ r \in \mathcal{R},\ \mathcal{K}_r = \left\{ k \in \overline{K}_r \ \middle|\ r \in \mathcal{R}_k \right\} \text{ and } \mathcal{K} = \bigcup_{r \in \mathcal{R}} \mathcal{K}_r.$$

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 $J \in \{2, \dots, |\mathcal{K}|\}$ and two sequences $(k_1, \dots, k_J) \in \mathcal{K}$ and $(r_1, \dots, r_J) \in \mathcal{R}$ 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) $\forall\ j \in \{1, \dots, J-1\}$, $r_j \in \mathcal{R}_{k_j} \cap \mathcal{R}_{k_{j+1}}$, *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 4. *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* $\displaystyle\sum_{k \in \overline{K}_r} d^*_{rk} \notin \mathbb{N}$.

Before proving Lemma 4, we assume this lemma holds and concludes the proof of Theorem 1. Therefore, using Lemma 4, we consider two sub-cases:

If Condition (*) holds, since $p \neq q$, then we know that there exists $J \in \{2, \dots, |\mathcal{K}|\}$ and two sequences $(k_1, \dots, k_J) \in \mathcal{K}$ and $(r_1, \dots, r_J) \in \mathcal{R}$ 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)* $\forall\ j \in \{1, \dots, J-1\}$, $r_j \in \mathcal{R}_{k_j} \cap \mathcal{R}_{k_{j+1}}$, and *(iv)* $r_J = q$ and $q \in \mathcal{R}_{k_J}$. Now consider

$$\epsilon = \min \left\{ \min_{j \in \{1, \dots, J-1\}} \left\{ d^*_{r_j k_j},\ 1 - d^*_{r_j k_j},\ d^*_{r_j k_{j+1}},\ 1 - d^*_{r_j k_{j+1}} \right\},\ d^*_{q k_J},\ 1 - d^*_{q k_J},\ d^*_{ql},\ 1 - d^*_{ql} \right\} > 0,$$

and $D^1 = \left(d^1, f^*\right)$ and $D^2 = \left(d^2, f^*\right)$ such that

$$\forall\ j \in \{1, \dots, J-1\},\ \begin{cases} d^1_{r_j k_j} = d^*_{r_j k_j} - \epsilon \\ d^1_{r_j k_{j+1}} = d^*_{r_j k_{j+1}} + \epsilon \end{cases} \qquad \forall\ j \in \{1, \dots, J-1\},\ \begin{cases} d^2_{r_j k_j} = d^*_{r_j k_j} + \epsilon \\ d^2_{r_j k_{j+1}} = d^*_{r_j k_{j+1}} - \epsilon \end{cases}$$

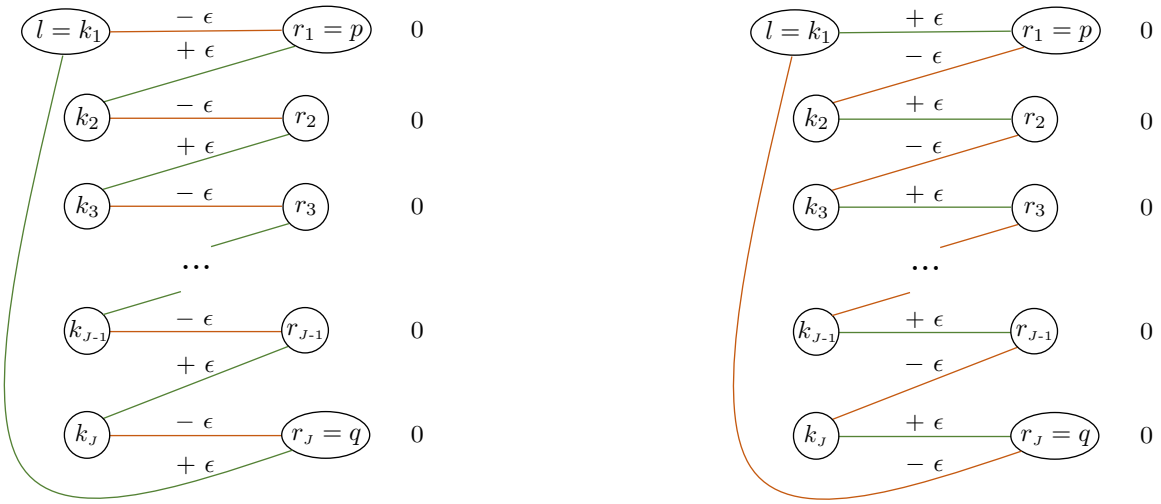
$$d^1_{q k_J} = d^*_{q k_J} - \epsilon \qquad d^2_{q k_J} = d^*_{q k_J} + \epsilon$$

$$d^1_{ql} = d^*_{ql} + \epsilon \qquad d^2_{ql} = d^*_{ql} - \epsilon$$

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 10. 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 $\forall k \in \{1, \dots, \overline{N}\}$, $\displaystyle\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}$. 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

$$\sum_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum_{r \in \overline{E}_{\nu_k}} d^*_{rk}.$$



(a) Illustration of D^1

(b) Illustration of D^2

Figure 10: 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 $(r_j)_{j\in\{1,\dots,J\}}$.

- 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 $\forall j \in \{1,\dots,J-1\}$, $r \neq r_j$ 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 $\exists j \in \{1,\dots,J-1\}$, $r = r_j$, 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 $d^* = \frac{1}{2}\left(d^1 + d^2\right)$ so $D^* = \frac{1}{2}\left(D^1 + D^2\right)$, 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 4 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 $\forall (j,i) \in \{1,\dots,J\} \times \{1,\dots,I\}$, $s_j \neq t_i$ and $k_j \neq l_i$. We now consider

$$\epsilon = \min \left\{ \begin{array}{c} \min\limits_{j \in \{1,\dots,J-1\}} \left\{ d^*_{s_j k_j},\ 1 - d^*_{s_j k_j},\ d^*_{s_j k_{j+1}},\ 1 - d^*_{s_j k_{j+1}} \right\},\ d^*_{s_J k_J},\ 1 - d^*_{s_J k_J}, \\ \min\limits_{i \in \{1,\dots,I-1\}} \left\{ d^*_{t_i l_i},\ 1 - d^*_{t_i l_i},\ d^*_{t_i l_{i+1}},\ 1 - d^*_{t_i l_{i+1}} \right\},\ d^*_{t_I l_I},\ 1 - d^*_{t_I l_I}, \\ \left(\overline{z_s} - \underline{z_s}\right) f^*_{s\underline{z_s}},\ \left(\overline{z_s} - \underline{z_s}\right)\left(1 - f^*_{s\underline{z_s}}\right),\ \left(\overline{z_s} - \underline{z_s}\right) f^*_{s\overline{z_s}},\ \left(\overline{z_s} - \underline{z_s}\right)\left(1 - f^*_{s\overline{z_s}}\right), \\ \left(\overline{z_t} - \underline{z_t}\right) f^*_{t\underline{z_t}},\ \left(\overline{z_t} - \underline{z_t}\right)\left(1 - f^*_{t\underline{z_t}}\right),\ \left(\overline{z_t} - \underline{z_t}\right) f^*_{t\overline{z_t}},\ \left(\overline{z_t} - \underline{z_t}\right)\left(1 - f^*_{t\overline{z_t}}\right), \end{array} \right\} > 0.$$

and $D^1 = \left(d^1, f^1\right)$ and $D^2 = \left(d^2, f^2\right)$ such that

$$\forall\ j \in \{1,\dots,J-1\},\ \left\{ \begin{array}{c} d^1_{s_j k_j} = d^*_{s_j k_j} - \epsilon \\ d^1_{s_j k_{j+1}} = d^*_{s_j k_{j+1}} + \epsilon \end{array} \right. \qquad \forall\ j \in \{1,\dots,J-1\},\ \left\{ \begin{array}{c} d^2_{s_j k_j} = d^*_{s_j k_j} + \epsilon \\ d^2_{s_j k_{j+1}} = d^*_{s_j k_{j+1}} - \epsilon \end{array} \right.$$

$$\forall\ i \in \{1,\dots,I-1\},\ \left\{ \begin{array}{c} d^1_{t_i l_i} = d^*_{t_i l_i} + \epsilon \\ d^1_{t_i l_{i+1}} = d^*_{t_i l_{i+1}} - \epsilon \end{array} \right. \qquad \forall\ i \in \{1,\dots,I-1\},\ \left\{ \begin{array}{c} d^2_{t_i l_i} = d^*_{t_i l_i} - \epsilon \\ d^2_{t_i l_{i+1}} = d^*_{t_i l_{i+1}} + \epsilon \end{array} \right.$$

$$d^1_{sk_J} = d^*_{sk_J} - \epsilon \qquad d^2_{sk_J} = d^*_{sk_J} + \epsilon$$

$$d^1_{tl_I} = d^*_{tl_I} + \epsilon \qquad d^2_{tl_I} = d^*_{tl_I} - \epsilon$$

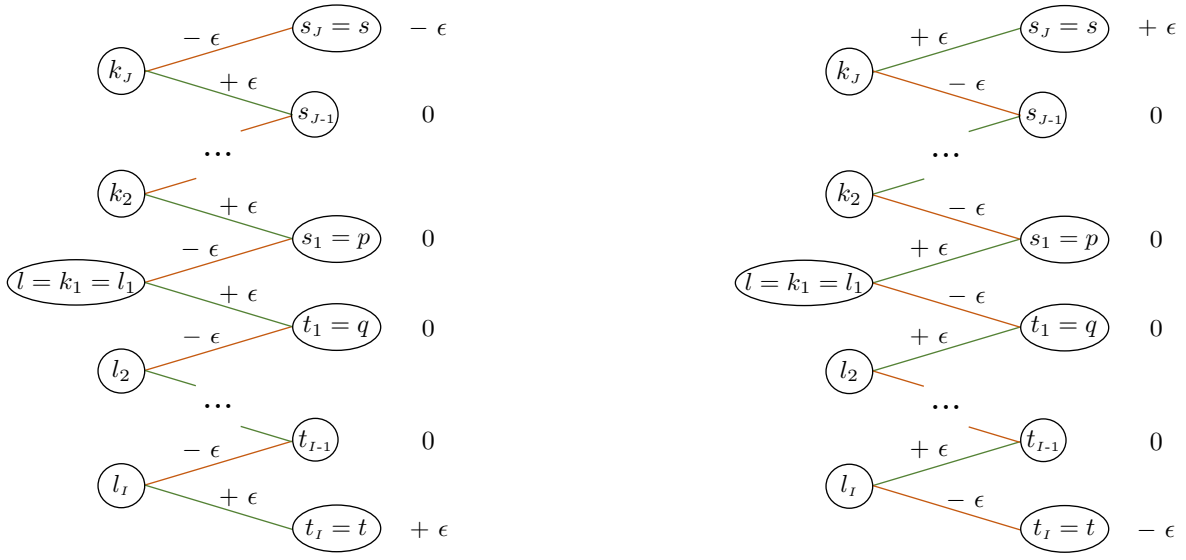
$$f^1_{s\underline{z_s}} = f^*_{s\underline{z_s}} + \frac{\epsilon}{\overline{z_s} - \underline{z_s}} \qquad f^2_{s\underline{z_s}} = f^*_{s\underline{z_s}} - \frac{\epsilon}{\overline{z_s} - \underline{z_s}}$$

$$f^1_{s\overline{z_s}} = f^*_{s\overline{z_s}} - \frac{\epsilon}{\overline{z_s} - \underline{z_s}} \qquad f^2_{s\overline{z_s}} = f^*_{s\overline{z_s}} + \frac{\epsilon}{\overline{z_s} - \underline{z_s}}$$

$$f^1_{t\underline{z_t}} = f^*_{t\underline{z_t}} - \frac{\epsilon}{\overline{z_t} - \underline{z_t}} \qquad f^2_{t\underline{z_t}} = f^*_{t\underline{z_t}} + \frac{\epsilon}{\overline{z_t} - \underline{z_t}}$$

$$f^1_{t\overline{z_t}} = f^*_{t\overline{z_t}} + \frac{\epsilon}{\overline{z_t} - \underline{z_t}} \qquad f^2_{t\overline{z_t}} = f^*_{t\overline{z_t}} - \frac{\epsilon}{\overline{z_t} - \underline{z_t}}$$

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 11.



(a) Illustration of D^1

(b) Illustration of D^2

Figure 11: 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 $(t_i)_{i \in \{1,\dots,I\}}$.

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 $\forall k \in \{1,\dots,\overline{N}\}$, $\sum\limits_{r \in \overline{E}_{\nu_k}} d^1_{rk} = \sum\limits_{r \in \overline{E}_{\nu_k}} d^2_{rk} = \sum\limits_{r \in \overline{E}_{\nu_k}} d^*_{rk}$. 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

$$\sum_{r\in\overline{E}_{\nu_{k_j}}} d^1_{rk_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^1_{rk_j} + d^1_{r_jk_j} + d^1_{r_{j-1}k_j} = \sum_{\substack{r\in\overline{E}_{\nu_{k_j}}\\ r\neq r_j, r_{j-1}}} d^*_{rk_j} + d^*_{r_jk_j} - \epsilon + d^*_{r_{j-1}k_j} + \epsilon = \sum_{r\in\overline{E}_{\nu_{k_j}}} d^*_{rk_j},$$

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 $\left(d^1, f^1\right)$ and $\left(d^2, f^2\right)$ verify the two types of constraints of $\overline{\mathcal{L}}(w)$ associated with r:

- If $r \neq s, t$, then $\forall z \in \{\widetilde{z}_r, \dots, Z\}$, $f^1_{rz} = f^2_{rz} = f^*_{rz}$. 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.

  - $\diamond$ If $\forall j \in \{1,\dots,J-1\}$, $r \neq r_j$ and $\forall i \in \{1,\dots,I-1\}$, $r \neq t_i$, then $\forall\ k \in \overline{K}_r$, $d^1_{rk} = d^2_{rk} = d^*_{rk}$, thus

$$\sum_{k\in\overline{K}_r} d^1_{rk} = \sum_{k\in\overline{K}_r} d^2_{rk} = \sum_{k\in\overline{K}_r} d^*_{rk}.$$

  - $\diamond$ If $\exists j \in \{1,\dots,J-1\}$, $r = s_j$, 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

$$\sum_{z\in\{\widetilde{z_s},\dots,Z\}} f^1_{sz} = \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^1_{sz} + f^1_{s\underline{z_s}} + f^1_{s\overline{z_s}} = \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^*_{sz} + f^*_{s\underline{z_s}} + \frac{\epsilon}{\overline{z_s}-\underline{z_s}} + f^*_{s\overline{z_s}} - \frac{\epsilon}{\overline{z_s}-\underline{z_s}} = \sum_{z\in\{\widetilde{z_s},\dots,Z\}} f^*_{sz} = 1,$$

and

$$\sum_{z\in\{\widetilde{z_s},\dots,Z\}} f^2_{sz} = \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^2_{sz} + f^2_{s\underline{z_s}} + f^2_{s\overline{z_s}} = \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} f^*_{sz} + f^*_{s\underline{z_s}} - \frac{\epsilon}{\overline{z_s}-\underline{z_s}} + f^*_{s\overline{z_s}} + \frac{\epsilon}{\overline{z_s}-\underline{z_s}} = \sum_{z\in\{\widetilde{z_s},\dots,Z\}} f^*_{sz} = 1,$$

Moreover,

$$\begin{aligned}
\sum_{z\in\{\widetilde{z_s},\dots,Z\}} zf^1_{sz} - \sum_{k\in\overline{K}_s} d^1_{sk} &= \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} zf^1_{sz} + \underline{z_s}f^1_{s\underline{z_s}} + \overline{z_s}f^1_{s\overline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^1_{sk} + d^1_{sk_J}\right)\\
&= \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} zf^*_{sz} + \underline{z_s}f^*_{s\underline{z_s}} + \frac{\epsilon\underline{z_s}}{\overline{z_s}-\underline{z_s}} + \overline{z_s}f^*_{s\overline{z_s}} - \frac{\epsilon\overline{z_s}}{\overline{z_s}-\underline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^*_{sk} + d^*_{sk_J} - \epsilon\right)\\
&= \sum_{z\in\{\widetilde{z_s},\dots,Z\}} zf^*_{sz} - \sum_{k\in\overline{K}_s} d^*_{sk} = \widetilde{z_s}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\sum_{z\in\{\widetilde{z_s},\dots,Z\}} zf^2_{sz} - \sum_{k\in\overline{K}_s} d^2_{sk} &= \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} zf^2_{sz} + \underline{z_s}f^2_{s\underline{z_s}} + \overline{z_s}f^2_{s\overline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^2_{sk} + d^2_{sk_J}\right)\\
&= \sum_{\substack{z\in\{\widetilde{z_s},\dots,Z\}\\ z\neq\underline{z_s},\overline{z_s}}} zf^*_{sz} + \underline{z_s}f^*_{s\underline{z_s}} - \frac{\epsilon\underline{z_s}}{\overline{z_s}-\underline{z_s}} + \overline{z_s}f^*_{s\overline{z_s}} + \frac{\epsilon\overline{z_s}}{\overline{z_s}-\underline{z_s}} - \left(\sum_{\substack{k\in\overline{K}_s\\ k\neq k_J}} d^*_{sk} + d^*_{sk_J} + \epsilon\right)\\
&= \sum_{z\in\{\widetilde{z_s},\dots,Z\}} zf^*_{sz} - \sum_{k\in\overline{K}_s} d^*_{sk} = \widetilde{z_s}.
\end{aligned}$$

In conclusion, $D^1, D^2 \in \overline{\mathcal{L}}(w)$. However, it is clear that $d^* = \frac{1}{2}\left(d^1 + d^2\right)$ and $f^* = \frac{1}{2}\left(f^1 + f^2\right)$ so $D^* = \frac{1}{2}\left(D^1 + D^2\right)$, 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 1. □

Proof of Lemma 4. 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 $\forall\ k \in \overline{K}_r \setminus \mathcal{K}_r$, $d^*_{rk} \in \{0,1\}$. 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

$$\Theta = \sum_{r \in \mathcal{T}_{p,l}} \sum_{k \in \mathcal{K}_r} d^*_{rk} = \sum_{k \in \mathcal{K}_{p,l}} \sum_{r \in \{r' \in \mathcal{T}_{p,l} | k \in \mathcal{K}_{r'}\}} d^*_{rk} = \sum_{k \in \mathcal{K}_{p,l}} \sum_{r \in \mathcal{R}_k \cap \mathcal{T}_{p,l}} d^*_{rk}.$$

Let $k \in \mathcal{K}_{p,l} \setminus \{l\}$. Let us show that

$$\mathcal{R}_k \cap \mathcal{T}_{p,l} = \mathcal{R}_k.$$

In order 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 2, 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 $J \in \{2, \dots, |\mathcal{K}|\}$ and two sequences $(k_1, \dots, k_J) \in \mathcal{K}$ and $(r_1, \dots, r_J) \in \mathcal{R}$ 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 $I \in \{2, \dots, |\mathcal{K}|\}$ and two new sequences $(l_1, \dots, l_I) \in \mathcal{K}$, $(s_1, \dots, s_I) \in \mathcal{R}$ 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 $j \in \{2, \dots, J\}$ such that $r_j = s$, then consider $I = j \in \{2, \dots, |\mathcal{K}|\}$ and

  $$(l_1, \dots, l_I) = (k_1, \dots, k_j) \in \mathcal{K} \text{ and } (s_1, \dots, s_I) = (r_1, \dots, r_j) \in \mathcal{R}.$$

  Clearly, $(l_1, \dots, l_I)$ and $(s_1, \dots, s_I)$ verify properties *(i)*, *(ii)*, *(iii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I}$.

- ◇ If $r \notin \{r_2, \dots, r_J\}$ but there exists $j \in \{2, \dots, J\}$ such that $k_j = k$. In this case, consider $I = j \in \{2, \dots, |\mathcal{K}|\}$ and

  $$(l_1, \dots, l_I) = (k_1, \dots, k_j) \in \mathcal{K} \text{ and } (s_1, \dots, s_I) = (r_1, \dots, r_{j-1}, s) \in \mathcal{R}.$$

  Clearly, $(l_1, \dots, l_I)$ and $(s_1, \dots, s_I)$ verify properties *(i)*, *(ii)*, *(iii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I}$.

- ◇ If $r \notin \{r_2, \dots, r_J\}$ and $k \notin \{k_2, \dots, k_J\}$. Since $k \notin \{k_1, \dots, k_J\}$, then we must have $J < |\mathcal{K}|$. Consider $I = J + 1 \in \{2, \dots, |\mathcal{K}|\}$ and

  $$(l_1, \dots, l_I) = (k_1, \dots, k_J, k) \in \mathcal{K} \text{ and } (s_1, \dots, s_I) = (r_1, \dots, r_J, r) \in \mathcal{R}.$$

  Clearly, $(l_1, \dots, l_I)$ and $(s_1, \dots, s_I)$ verify properties *(i)*, *(ii)* and *(iv)* $s_I = s$ and $s \in \mathcal{R}_{l_I} = \mathcal{R}_k$. Note that $\forall\ i \in \{1, \dots, I-1\}$, property *(iii)* holds by definition of $(k_1, \dots, k_J)$ and $(r_1, \dots, r_J)$. 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 $\mathcal{K}_{p,l} \setminus \{l\} \subset \{1, \dots, \overline{N}\}$ 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 4. □

Theorem 2. *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 1. 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 $\forall\ k \in \overline{K}_r$, $d^*_{rk} \in \{0, \dots, 1\}$, 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,

$$\begin{aligned}
\sum_{z \in \{\widetilde{z_r}, \ldots, Z\}} \beta_z f_{rz} &= \sum_{\substack{z \in \{\widetilde{z_r}, \ldots, Z\} \\ z \neq \overline{z}, \underline{z}, z^*}} \beta_z f_{rz} + \beta_{\overline{z}} f_{r\overline{z}} + \beta_{\underline{z}} f_{r\underline{z}} + \beta_{z^*} f_{rz^*} \\
&= \sum_{\substack{z \in \{\widetilde{z_r}, \ldots, Z\} \\ z \neq \overline{z}, \underline{z}, z^*}} \beta_z f^*_{rz} + \beta_{\overline{z}} f^*_{r\overline{z}} - \frac{\epsilon}{\overline{z} - z^*}\beta_{\overline{z}} + \beta_{\underline{z}} f^*_{r\underline{z}} - \frac{\epsilon}{z^* - \underline{z}}\beta_{\underline{z}} + \beta_{z^*} f^*_{rz^*} + \left(\frac{\epsilon}{\overline{z} - z^*} + \frac{\epsilon}{z^* - \underline{z}}\right)\beta_{z^*} \\
&= \sum_{z \in \{\widetilde{z_r}, \ldots, Z\}} \beta_z f^*_{rz} - \frac{\epsilon\,(\overline{z} - \underline{z})}{(\overline{z} - z^*)\,(z^* - \underline{z})}\left(\frac{z^* - \underline{z}}{\overline{z} - \underline{z}}\beta_{\overline{z}} + \frac{\overline{z} - z^*}{\overline{z} - \underline{z}}\beta_{\underline{z}} - \beta_{z^*}\right).
\end{aligned}$$

Since γ verifies Condition (A), $z^*, \overline{z}, \underline{z} \in \{0, \ldots, Z\}$ and $\underline{z} < z^* < \overline{z}$, then using Lemma 2, 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 2. □

C Speed up of $\Lambda(v)$

Recall that $\Lambda(v)$ corresponds to

$$\min_{(w,D)} \left(c^T D\right) \qquad \text{Equation (20)}$$

$$s.t. \begin{cases} D \in \mathcal{D} & \text{Equations (7)-(12)} \\ (w, D) \in \mathcal{L} & \text{Equations (13)-(14)} \\ \sum_{s \in L_n} w_{ns\kappa_n} = 1 & \forall\ n \in \{1, \dots, \overline{N}\} \end{cases}$$

As we mention in Section 5, there are several variables that we can set up to 0. Among these, we have the four following ones:

$$\forall\ n, k \in \{1, \dots, \overline{N}\} \text{ s.t. } k \neq \kappa_n, \text{ then } \forall\ s \in L_n, w_{nsk} = 0.$$

$$\text{Let } n\{1, \dots, \overline{N}\} \text{ s.t. } \kappa_n = 1, \text{ then } \forall\ s \in \mathcal{S}^{(E)} \setminus L_n,\ d_s^i = 0.$$

$$\forall\ k \in \{1, \dots, \overline{N}\} \text{ s.t. } \kappa_n = k, \text{ then } \forall\ s \in \mathcal{S}^{(L)} \setminus L_n, \forall\ r \in \mathcal{S}^{(E)} \setminus E_n,\ d_{srk}^{(L)} = 0.$$

$$\forall\ k \in \{2, \dots, \overline{N}\} \text{ s.t. } \kappa_n = k - 1 \text{ and } \kappa_m = k, \text{ then } \forall\ r \in \mathcal{S}^{(E)} \setminus E_n, \forall\ s \in \mathcal{S}^{(L)} \setminus L_m,\ d_{rsk}^{(E)} = 0.$$